SIGT:UM SISTEMA INTELIGENTE PARA O GERENCIAMENTO DO TRÁFEGO FERROVIÁRIO

Raul Fonseca Neto

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO .

Aprovada por:

Prof. Antônio de Almeida Pinho, D.S.c
(Presidente)

Prof. Carlos Alberto da Silva Franco, D.Sc.

Prof. Cláudia Guerreiro Ribeiro do Vale, D.Sc.

Prof. Emmanuel Piseces Lopes Passos, Ph.D.

Prof. Nélson Maculan Filho, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
JUNHO DE 1990

FONSECA, RAUL NETO

SIGT: Um Sistema Inteligente para o Gerenciamento do Tráfego Ferroviário [Rio de Janeiro] 1990

XXIII, 523 p. 29.7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1990)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Inteligência Artificial I. COPPE/UFRJ

II. Título (série).

Ā minha espôsa Andréa

Aos meus filhos

Aos meus pais

Agradecimentos:

Meu particular agradecimento ao Prof. Antônio de Almeida Pinho, cuja orientação e confiança creditada à minha pessoa, foram determinantes na consolidação deste trabalho.

Ao Prof. Félix Mora-Camino, pela orientação do mestrado, pelo tema de tese e pela orientação inicial prestada no início do curso.

Aos Profs. Nélson Maculan Filho e Carlos Aberto Franco pela valiosa orientação dos trabalhos de qualificação.

Ao Programa de Engenharia de Sistemas e Computação, funcionários e professores, pelas condições de trabalho e de aprendizado oferecidas no decorrer do curso.

Aos companheiros Álvaro, Lorena, Vítor, Luíz Satoru, Luíz Torres, Adílson, Amorin, Ronaldo, Vermelho, Carlos Alberto, Valdir, André, entre outros, que sempre me incetivaram e contribuiram, no dia a dia, na solução de inúmeras tarefas.

Aos funcionários da Biblioteca Central, NCE, COPPE, cantinas e xerox, pela presteza e sinceridade no atendimento e convívio proporcionado ao longo dos vários anos de estudo e pesquisa.

Aos colegas de magistério da Universidade Federal de Juiz de Fora, que foram, sempre que necessário, solidários às minhas solicitações encaminhadas a Chefia do Departamento.

Aos colegas da RFFSA / SR-3 - Juiz de Fora, que contribuíram para a consolidação da pesquisa, respondendo a inúmeras questões, e permitindo o acesso, sem restrições, as funções de gerenciamento e controle do tráfego ferroviário;

À todos aqueles que, de uma forma ou outra , contribuíram para o sucesso deste trabalho.

Resumo da tese apresentada a COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.).

SIGT: UM SISTEMA INTELIGENTE PARA O GERENCIAMENTO DO TRÁFEGO FERROVIÁRIO

Raul Fonseca Neto

ORIENTADOR : Antônio de Almeida Pinho

PROGRAMA : Engenharia de Sistemas e Computação

Este trabalho apresenta o desenvolvimento teórico e fornece a implementação básica de um sistema de planejamento capaz de realizar o gerenciamento e controle de trens em uma ferrovia, contribuindo para a automação parcial e otimização do serviço de despacho.

Entre as características principais do sistema, podemos destacar a sua conduta objetiva e racional, voltado para a solução do problema, permitindo grande flexibilidade na realização de um planejamento dinâmico e interativo, utilizando, para tanto, o conhecimento declarativo e heurístico, associado ao domínio da tarefa.

O sistema foi construído com o uso combinado de técnicas de Inteligência Artificial, Teoria de Sistemas, Pesquisa Operacional e Micro-Computação, tendo como hipotéses fundamentais a viabilidade de se realizar a operação de trens na malha ferroviária com o uso de planejamento e a possibilidade de se modelar matematicamente o problema de despacho de trens.

Mais precisamente, formulamos um problema de programação multiobjetiva "fuzzy", que utiliza um conjunto de funções de pertinência que avaliam o nível de satisfação dos objetivos e incorpora as restrições estruturais do problema de "scheduling", determinando uma

representação espaço-estado e a definição de um sistema fisíco simbólico que permitiu a geração de planos de horários de forma eficiente.

Dando continuidade, desenvolvemos uma arquitetura que suporta o gerenciamento inteligente de planos de ações ou a tomada de decisões, incluindo as funções básicas de planejamento, replanejamento, monitoração, otimização e simulação.

A experimentação do sistema foi possível através da implementação de um modulo que permite a simulação da execução do planejamento, atraves da monitoração e consequente avaliação de planos na presença de operações simuladas. Tais experimentos, viabilizaram, a primeira vista , a implementação de um sistema de gerenciamento em tempo real, segundo um modelo de monitoração e replanejamento.

O programa de computador foi desenvolvido na linguagem de programação Modula-2, segundo as técnicas mais avançadas de projeto e construção, sob o enfoque de múltiplos paradígmas, explorando as características necessárias ao desenvolvimento de sistemas de programação em Inteligência Artificial.

Abstract of thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degres of Doctor of Science (D. Sc.).

AN INTELLIGENT SYSTEM FOR THE RAILWAY TRAFFIC MANAGEMENT

Raul Fonseca neto

Thesis Supervisor: Antônio de Almeida Pinho

Department: Engenharia de Sistemas e Computação

This dissertation deals with the theoretical development and computational aspects of a planning system which is able to realize the railway traffic management, providing the parcial automation and optimization of the dispatching service.

Among the main features of the system, we can distinguish its objective and rational goal, directed to the problem solving, allowing a great flexibility in making interactive and dynamic planning. For this purpose the system uses a declarative and heuristic knowledgement, associated with the task domain.

The system was developped with the combined use of techinics related to Artificial Intelligence, Operations Research, System Theory and Micro-Computation. The fundamental hypothesis were the viability of making the train operation in a railway line with the use of planning and the possibility of formulating a mathematical model to the train

dispatching problem.

In other words, we formulated a fuzzy multi-objective programming problem that uses a set of membership functions which evaluates the satisfaction level of the goals and incorporates the structured constraints of the scheduling problem , defining a state-space representation and a symbolic phisical system wich allows the generation of schedules plans in a efficient way.

Indeed, we developped an intelligent architecture that supports the management of plans of actions or the decision-making process , including the basic functions of replanning, monitoring, optimization and simulation.

The system experiment was possible by the use of a module implementation wich let a simulated execution through the monitoring process that evaluates the planning feasibility. These experiments provided, at first sight, the implementation of a real-time management system , from a monitoring and replanning model.

The software was developped in programming language Modula-2 with the most advanced techinics and exploring the necessary features related to Artificial Intelligence programming systems.

Organização dos Capítulos: Página:

Conteúdo: Página:

Capítulo I
Introdução

1.1 - Apresentação, Justificativas e Objetivos :

O transporte ferroviário tem, através dos tempos, contribuido para o desenvolvimento econômico e social das nações. O sistema ferroviário, que possibilita a realização deste transporte, e extremamente complexo comprendendo vários sub-sistemas que se interagem com o objetivo principal de servir ao deslocamento de cargas e passageiros. Entre os sub-sistemas mais importantes, destacamos o sub-sistema operacional , compreendendo os servicos de gerenciamento e controle do tráfego de trens na malha ferroviária.

A crescente automatização do meio ferroviário, incluíndo os sistemas de sinalização e controle de trafégo, encorajou-nos a desenvolver um sistema, denominado SIGT (sistema inteligente para o gerenciamento do tráfego ferroviário), que permitisse a realização do gerenciamento e controle de trens, de forma eficiente, sob a ótica do planejamento, visando a automação parcial deste serviço na ferrovia.

Desta forma, apresentamos neste trabalho, um estudo teórico que culminou no desenvolvimento de um programa de computador capaz de gerenciar o tráfego ferroviário. Mais precisamente, consideramos uma proposta de automação parcial do serviço de despacho de trens em um trecho da ferrovia, constituído de seções de linha dupla ou singela sob controle de tráfego centralizado, possibilitando, com o gerenciamento em tempo real, uma ganho, estimado em 25%, em relação a capacidade de tráfego existente.

O objetivo principal de nossa proposta é, como vimos, otimizar o serviço de despacho de trens, considerando, para tanto, a resolução de sucessivos conflitos , de uma forma sistemática, atendento as condições de segurança, e possibilitando o desenvolvimento de linhas de raciocínio.

Para tanto, consideramos o desenvolvimento de um sistema de planejamento que procura efetuar o controle operacional através do gerenciamento inteligente de um plano de horários convertido em um plano de ações que determinam a solução dos futuros conflitos.

Entre as principais características do sistema, podemos destacar a sua conduta racional e objetiva, voltada para a solução do problema de sequenciamento de trens, colocado sob a forma de um problema de planejamento e "scheduling", permitindo grande flexibilidade na realização de um planejamento dinâmico e interativo com a utilização intensiva do conhecimento de especialistas.

O sistema foi construído com o uso combinado de técnicas de Inteligência Artificial, Teoria de Sistemas e Micro-Computação, tendo como hipóteses fundamentais a possibilidade de modelarmos matematicamente o problema de sequenciamento de trens como um sistema dinâmico baseado na ocorrência de eventos discretos, na determinação de uma representação espaco-estado, que extende a modelagem matemática, e, finalmente, no desenvolvimento de um sistema simbólico associado ao conhecimento intensivo, que formam a base para a estruturação de um sistema de planejamento , dependente do domínio, que permitiu a realização do gerenciamento do tráfego ferroviário.

A presente implementação incorpora o embasamento teórico de uma arquitetura que suporta o gerenciamento inteligente de planos de ações, ou a tomada de decisões no controle operacional, incluindo as funções básicas de planejamento, replanejamento, simulação, otimização e monitoração. Neste sentido, o programa de computador foi desenvolvido segundo as técnicas mais avançadas de projeto, explorando as características necessárias ao desenvolvimento de sistemas de programação em Inteligência Artificial.

1.2 - Organização e Limitações:

Quanto a organização do trabalho, após este primeiro capítulo introdutório, apresentamos o segundo capítulo que descreve, inicialmente, as características básicas de sistemas inteligentes e mostra a evolução dos sistemas de gerenciamento observando a utilização de técnicas de Inteligência Artificial, Micro-Computação e Teoria de Sistemas. Em seguida, fazemos uma descrição geral de nosso sistema, quanto a sua concepção, arquitetura e capacidade funcional. Na sequência, justificamos a viabilidade e necessidade do uso de planejamento no controle da operação de trens e, finalmente, destacamos

a importância do sistema apresentando a sua aplicabilidade em diferentes fases do planejamento ferroviário.

Mais adiante, no capítulo III, descrevemos os aspectos teóricos de um modelo voltado para a geração de planos de horários, que determinam as ações necessárias ao gerenciamento do tráfego ferroviário. Do ponto de vista matemático, desenvolvemos um modelo de programação multi-objetiva "fuzzy", para o problema de sequenciamento de trens, contendo um conjunto de restrições estruturais do problema de "scheduling" e um conjunto de restrições flexíveis associadas aos objetivos. O algoritmo proposto para a solução deste modelo, e baseado na realização de uma busca heurística no espaço de estados do problema, segundo uma heurística admissível que comprende um conjunto de funções de pertinência que avaliam o nível de satisfação dos objetivos.

No capitulo IV, em primeiro lugar, abordamos a viabilidade da utilização de sistemas baseados em conhecimento, ou sistemas de conhecimento intensivo, em tarefas de planejamento e "scheduling" e apresentamos as principais formas de representação do conhecimento e metódos de raciocínio em Inteligência Artificial. Em seguida , mostramos o desenvolvimento de um sistema baseado em conhecimento, independente do domínio, que combina a utilização de sistemas de produçao com estruturas de representação do conhecimento na forma de "frames". A implementação deste sistema se baseia na filosofia de tipos abstrato de dados e na técnica de decomposição funcional, sendo projetado com a finalidade de suportar a realização de inferências pelo sistema de planejamento, determinando ações satisafatórias compatíveis com o sistema voltado para o gerenciamento do tráfego de trens.

No capitulo V, descrevemos o conhecimento intensivo utilizado pelo sistema, ressaltando a representação do conhecimento declarativo referente a atributos e relacionamentos dos objetos e entidades do sistema ferroviário, e o conhecimento heurístico e imperativo referente as restrições e comandos que regem o processo decisório responsavél pelo gerenciamento do tráfego de trens. Apresentamos, tambem, o processo de desenvolvimento de novas heurísticas, a partir da análise de situações específicas de conflitos com o uso de um modelo de simulação.

No capitulo VI, discutimos nosso sistema segundo a visão de

arquiteturas inteligentes, ressaltando a integração de sistemas baseados em conhecimento com as hipóteses fundamentais da Inteligência Artificial, permitindo o uso combinado de técnicas de busca, raciocínio e satisfação de restrições na solução do problema de geração de planos. Este processo consiste, essencialmente, na atribuição de preferência a operadores, durante o processo de busca, reduzindo o espaço de soluções de problemas. Em seguida, abordamos as questões fundamentais que delineam o desenvolvimento de sistemas inteligentes, como o uso combinado de técnicas de Inteligência Artificial e Pesquisa Operacional bem como do processo universal de decomposição. Finalmente, apresentamos neste capitulo, a base teórica de um modelo de simulação discreta, que considera a filosofia de simulação por processos e exame de atividades, com o uso de técnicas de Inteligência Artificial, culminando no desenvolvimento de um programa de simulação que apresenta alto grau de paralelismo na realização de eventos. Este programa, que consiste de uma rotina do sistema SIGT, foi de fundamental importância para a validação do sistema baseado em conhecimento e desenvolvimento de novas heuristicas, como mencionamos anteriormente.

No capitulo VII , apresentamos o desenvolvimento teórico de nosso sistema segundo o enfoque da teoria de planejamento, envolvendo questões como: representação de planos e ações, planejamento linear, planejamento nao-linear e hierárquico, estratégias ou processos de desenvolvimento de planos, dominância entre planos, planejamento e otimização, paralelismo entre ações, planejamento com incerteza, planejamento temporal, organização de sistemas de planejamento, credibilidade de planos e, por fim, execução de planos. Finalizando, apresentamos a implementação presente de nosso sistema que permite, através da simulação do planejamento, em condições próximas as condições reais de operação, verificar a viabilidade de um sistema de monitoração e replanejamento que permitira a execução do planejamento em tempo real.

Finalmente, no capitulo VIII, sao apresentadas as conclusões resultantes do desenvolvimento de nosso trabalho. Mostramos, tambem, o que nao foi possível ser feito, determinando, assim, sugestões para o desenvolvimento de futuros estudos nesta direção.

Neste sentido, queremos ressaltar que o trabalho carece de resultados práticos, dado a complexidade e dificuldades que, certamente,

acompanham uma possível implementação do sistema em tempo real. No entanto, o desenvolvimento teórico apresentado justifica , desde já, uma metodologia básica que propiciará a implementação futura de sistemas de gerenciamento inteligente do tráfego ferroviário em condições reais de operação.

Após as referências bibliográficas, apresentamos um apêndice que propõe uma classificação resumida de nosso sistema, destacando as principais características incorporadas ao mesmo, segundo a modelagem matemática, métodos de solução, formas de representação do conhecimento, métodos de raciocínio, técnica de simulação e sistema de planejamento utilizados. Consideramos, também, um segundo apêndice, onde apresentamos os módulos de especificação do programa, que definem as interfaces do sistema, e os diagramas de fluxo de controle entre módulos, incluíndo as funções principais.

Quanto a forma da redação, queremos considerar o fato de utilizarmos, ao longo do texto, algumas expressões e termos técnicos da língua inglesa, sem a devida traduçao. Tal procedimento se justifica dado a inexistência de traduções consagradas em nossa literatura técnica.

Como exemplo, podemos citar o uso das expressões: "scheduling", com o significado de programação ou escalonamento de horários; "fuzzy", com o significado de adjetivo difuso ou nebuloso; e, finalmente, "frame", com o significado de substantivo que representa a existência de uma estrutura de representação e organização de conhecimento.

Capítulo II

Descrição do Sistema SIGT Segundo a Concepção de Sistemas Inteligentes

2.1 - Inteligência Artificial e Sistemas de Suporte a Decisões: Integração e Evolução

O objetivo principal desta seção é apresentar uma abordagem clara e concisa de sistemas inteligentes, bem como, esboçar uma análise da evolução dos sistemas de suporte a decisões face, a integração de tais sistemas com técnicas de Intéligencia Artificial.

Mostramos, também, a potencialidade dos sistemas inteligentes para o exercício das funções de gerenciamento e controle através do uso combinado de técnicas de Inteligência Artificial, Teoria de Sistemas e Micro-Computação.

2.1.1 - Sistemas Inteligentes:

- Conceituação e Propriedades:

Primeiramente, vamos tentar caracterizar os principais atributos de sistemas inteligentes, seguindo uma breve descrição apresentada por (A. Newell [1980]). Segundo Newell um sistema inteligente deve ser capaz de :

- operar em tempo real;
- explorar conhecimento;
- utilizar símbolos e abstrações;
- ser tolerável a falhas e imprevistos;
- comunicar-se em linguagem natural;
- ter capacidade de aprendizado e, finalmente,
- ter uma conduta objetiva voltada para a solução de problemas.

Indubitavelmente, um sistema com a finalidade de dar apoio ou orientar decisões relativas ao planejamento estratégico, tático e ao controle operacional de uma ferrovia deve, na medida do possível, suportar as condicionantes descritas por Newell.

Como veremos adiante, existem diferenças marcantes entre os sistemas mais primitivos e os sistemas atuais denominados inteligentes.

A evolução e consequente maturação destes sistemas , voltados ao controle da gestão, ou ao gerenciamento de sistemas complexos, não estruturados e imprecisos, surge da funcionalidade e das interrelações com o meio externo no qual os mesmos se inserem.

Dentro deste ponto de vista, podemos considerar os sistemas de informação para gerenciamento, como sistemas primitivos com finalidade quase exclusiva de prover informações ao tomador de decisões , não sendo, portanto, capazes de possuírem uma conduta objetiva, voltada para a solução de problemas .

Entretanto, se considerarmos um sistema que, efetivamente, suporte o gerenciamento de trens, teremos que observar alguns aspectos relativos a funcionalidade e a sua interrelação com o sistema ferroviário, principalmente, no que tange a operação de trens.

É nossa proposta, oferecer um sistema que permita suportar as funções básicas de planejamento, monitoração e replanejamento do tráfego ferroviário, objetivando, principalmente, a otimização e automação parcial do serviço de despacho de trens.

Infelizmente, no que diz respeito aos atributos de sistemas inteligentes, não abordamos, com profundidade, as questões relativas a capacidade de aprendizado automático e o desenvolvimento de uma interface em linguagem natural. Entretanto, o sistema apresentado caracteriza-se por uma conduta objetiva e racional possuindo grande flexibilidade, explora vasta quantidade de conhecimento na tarefa de solução de problemas e constitui um sistema físico simbólico, incorporando desta forma os principais atributos necessários a tomada de decisões inteligentes, segundo Raj Reddy [1988].

A. Newell e H. Simon [1976], no trabalho vencedor do prêmio Turing de 1975, promovido pela Association of Compunting Machinery , definem uma lei de natureza qualitativa para sistemas simbólicos, mais conhecida como hipóteses do sistema físico simbólico, segundo a qual estes sistemas tem condição necessária e suficiente para a tomada de decisões inteligentes.

Segundo Newell e Simon , sistemas simbólicos são coleções de padrões e processos, nos quais os processos são capazes de produzir, destruir ou modificar os padrões. A propriedade mais importante dos padrões e que eles podem designar objetos, processos ou outros padrões e

quando designam processos, podem ser interpretados e, consequentemente, dado algum padrão, podemos invocar o processo que o mesmo designa.

O adjetivo físico para os sistemas simbólicos denota, claramente, que tais sistemas obedecem as leis da física e não se restrigem, apenas, a sistemas simbólicos humanos. Um sistema físico simbólico tem a capacidade de solucionar problemas, o que representa uma indicação primária de inteligência, gerando e progressivamente modificando estruturas simbólicas até produzir uma estrutura solução. O processo de geração e modificação de estruturas e caracterizado como uma busca inteligente que emprega o conhecimento a respeito do domínio do problema , e é conhecido como a hipótese de busca heurística.

No planejamento do tráfego ferroviário, este domínio pode ser descrito com todas as possíveis alternativas viáveis de operação ou despacho de trens e seu conhecimento afim, envolvendo restrições de movimento, rota de trens, estratégias de operação, "lay-out" da malha ferroviária, regras de sinalização , etc.

2.1.2 - Hierarquia dos Sistemas de Gerenciamento:

- "Management Information Systems" (MIS):

Os sistemas de informação ao gerenciamento (MIS) oferecem ao tomador de decisões um conjunto de informações necessárias a realização do planejamento estratégico, tático e operacional pretendido. Estes sistemas são, fundamentalmente, embasados na teoria de banco de dados e de sistemas de informação , oferecendo a possibilidade de consulta e atualização "on line" das informações estruturadas em uma topologia e abordagem clássica de banco de dados (C.J. Date [1986]).

E de grande importância o projeto da interface destes sistemas, que especifica as facilidades de comunicação e acesso existentes na interrelação com o usuário. Atualmente, é possível a utilização de uma linguagem de manipulação de dados mais apropriada, possuindo um dicionário específico que facilita a interação por meio de perguntas aproximando-se da interface em linguagem natural.

Uma das maiores restrições dos sistemas de informação, apoiados

na teoria de banco de dados, esta na necessidade de rapidez das operações de atualização e consultas , que devem ser compatíveis com a aplicação afim. Estas operações, são basicamente operações de manipulação de arquivos apoiadas em técnicas de estruturas de dados e algorítmos convencionais. Neste sentido, tem-se apresentado novas abordagens na toria de banco de dados, incluindo o desenvolvimento do modelo entidade-relacionamento (P. Pin-Shan Chen [1976]), de sistemas de banco de dados orientados para objetos (R. King [1986]), integração de bases de conhecimento e banco de dados (Ye-Sho Chen [1988]) e o emprego de técnicas de Inteligência Artificial (G.P. Zarri [1986]).

- "Decision Support Systems" (DSS):

Os sistemas de suporte a decisão (DSS) são sistemas que auxiliam o tomador de decisões nas questões relativas ao planejamento estratégico, tático e operacional. Segundo a definição mais apropriada, sugerida por (M.J. Ginzberg e E.A. Stohr [1982]), DSS são sistemas de informação baseados em computador utilizados para suportar a tomada de decisões em situações onde não e possivel ou desejável ter-se um sistema completo e autônamo que se encarregue de todo o processo de decisão.

Alguns conceitos são de grande importância na definição dos DSS , entre os quais:

. tipo de problema e funções suportáveis (G.A. Gorry e M.S. Morton [1971]);

. função do sistema e características de interface (J.D. Little [1970]);

. objetivos e finalidades do sistema (S.L. Alter [1980]);

. capacidades e finalidades do sistema (J.H. Moore e M.G. Chang [1980]);

. componentes do sistema (P.H. Bonczeck e outros [1980]) e

. processo de desenvolvimento (P.G.W. Keen [1980]).

Entre as características principais que um DSS deve possuir podemos citar:

. ser simples e robusto;

. ser de fácil controle;

- ser adaptativo;
- ser completo nas questões afins e
- ser de fácil comunicação.

Moore e Chang [1980], seguindo os conceitos de capacidade e finalidade dos sistemas , colocam como pontos principais a capacidade dos DSS suportarem análise e modelagem de dados "on line" e serem orientados para planejamento futuro ou estratégico .

Para terem esta capacidade de análise e modelagem de dados ,estes sistemas utilizam teorias clássicas de Pesquisa Operacional, Análise Econômica e Análise Estatística. Portanto, estes sistemas fazem uso de modelos e métodos computacionais e incorporam algumas técnicas bastantes conhecidas na literatura, como por exemplo: programação linear, análise custo/benefício, modelos de regressão, métodos probabilísticos, teoria de utilidade, etc.

Segundo Hebert Simon [1981], o campo de técnicas computacionais não se reduz, necessariamente, a otimização. Ao contrário, os métodos tradicionais de engenharia e outras ciências fazem muito mais uso de soluções satisfatórias que atendam à níveis de aspiração e sejam, consequentemente, racionais.

Para a descoberta de soluções ou ações satisfatórias, os DSS empregam técnicas heurísticas. Para conceituarmos heurística, utilizaremos uma definição clássica de E. A. Feigenbaum e J. Feldman [1963] , que classificam heurística como:

" uma estratégia, pensamento, tentativa ou algo que drasticamente limite a busca de soluções em problemas complexos. A heuristica não garante uma solução ótima, nem garante, sempre, uma solução. Uma boa heuristica pode ser considerada como aquela que oferece boas soluções quando precisamos, ou na medida do possível."

Podemos, também, utilizar a definição de J. Pearl [1984]. Segundo Pearl :

" heuristica é algum critério, método ou princípio para decidir qual entre várias alternativas promissoras, é aquela que tenha conduta mais objetiva. Ela representa um compromisso entre dois requerimentos: ser mais simples o possível e, ao mesmo tempo, discriminar corretamente

soluções boas e ruins."

Para, D. N. Chorafas [1987] um sistema de suporte a decisões (DSS) pode possuir algumas regras de conhecimento e um diálogo interativo com o usuário. Entretanto, a capacidade de sugerir e avaliar situações, empregando raciocinio e vasta quantidade de conhecimento na forma declarativa, são características de sistemas especialistas que representam uma evolução natural dos sistemas de suporte a decisões.

Walter Reitman [1982] discute a aplicação de técnicas de InteligÊncia Artificial em DSS, examinando a possibilidade destes sistemas fornecerem boas alternativas através de uma busca restrita.

Dentre os DSS que empregam esta visão alternativa, podemos destacar o sistema ARIADNE (Alternative Ranking Interactive Aid based on Dominance Structural Information Elicitation) apresentado por (A.P.Sage e C.C. White [1984]). Este sistema admite um processo de decisão e planejamento interativo em situações onde existem múltiplos critérios de seleção de alternativas nas quais os valores de pontuação minimos e os valores das utilidades esperadas , nos casos de decisão sob risco, sejam conhecidos.

- "Expert Systems" (ES):

E.A. Feigenbaum e outros [1981], definem sistemas especialistas como:

" programas inteligentes de computador que utilizam conhecimento e procedimentos de inferência para resolverem problemas que requerem um especialista para a sua solução ".

O conhecimento de um sistema especialista e constituído de fatos e heurísticas. Os fatos são a informação considerada de conhecimento público, enquanto que, as heurísticas, são informações de especialistas que caracterizam o grau de esperteza das decisões .

A arquitetura básica de um sistema especialista e mostrada na figura 1 (P. Harmon e D. King [1985]).

A estratégia de inferência mais comumente utilizada em sistemas especialistas ou sistemas de conhecimento, e a aplicação da regra lógica heuristica) na forma :

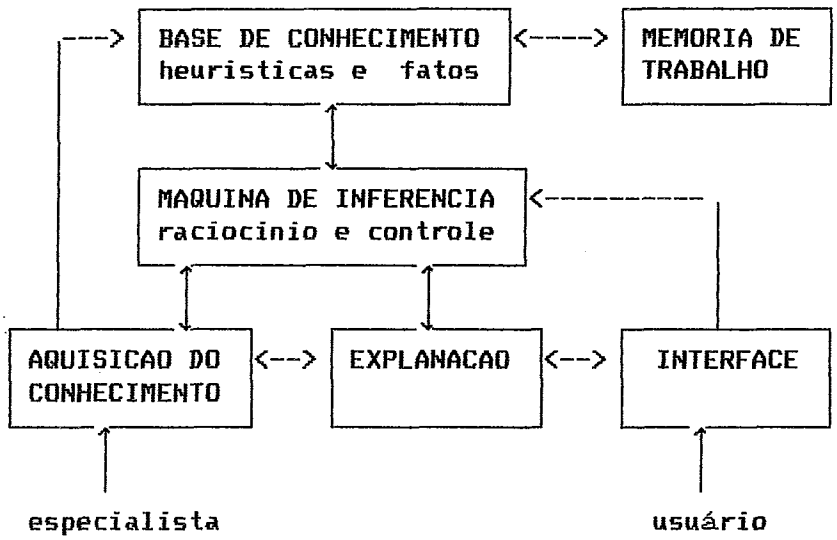


figura 1: arquitetura básica de um sistema especialista.

Se condição Então ação ,

e soubermos que a condição (antecedente da regra) é verdadeira, podemos empregar a ação desejada (consequente da regra).

A estrutura descrita acima e conhecida como regra de produção, e o sistema especialista que emprega o mecanismo de inferência para a execução de regras baseado na aplicação do "modus ponens" é conhecido como sistema de produção (F.Hayes-Roth [1985]). Os novos fatos gerados pela execução das regras recebem o nome de elementos da memória de trabalho, possuindo um forte paralelo com a memória de curto prazo segundo a forma de raciocínio humano descrito por A. Newell e H. Simon [1972] .

Este tipo de sistema dispõem de uma forma de controle (encadeamento de regras) que pode ser orientado segundo os fatos ("forward chaining") ou orientado segundo os objetivos ("bacward chaining"); em ambas as formas, o encadeamento procura estabelecer um caminho entre os dados conhecidos e os objetivos desejados.

As principais diferenças dos sistemas especialistas (programação simbólica) em relação aos sistemas tradicionais (

programação convencional) são mostradas no quadro 1 :

quadro 1: diferencas entre programação simbólica e programação convencional:

Programação Simbolica	Programação Convencional
- utiliza heurísticas	- utiliza algoritmos
- orientado para processamento simbólico	- orientado para processamento numérico
- processamento interativo	- processamento sequencial
- explanação de raciocínio	- não há explanação
- controle não determinístico	- controle determinístico
- utiliza base de conhecimento	- utiliza estrutura de dados
- conhecimento declarativo	- conhecimento procedural
- fácil manutenção	- difícil manutenção

Existem, nos sistemas atuais, outras formas importantes para representação do conhecimento de longo termo (fatos) entre as quais podemos destacar a utilização de "frames" (R.Fikes e T. Kelhler [1985]) que permitem o desenvolvimento de estruturas com propriedades de especialização (" is a"), composição ("part of") e instanciação ("instance of").

- "Decision Support / Decision Simulation" (DS/DSIM)

O objetivo dos sistemas de suporte a decisão é, basicamente, assistir a análise de atividades relacionadas a tarefas de tomada de decisões, como por exemplo controle e planejamento.

Entretanto, para que tais decisões sejam bem assistidas e necessário que o sistema modele ou idealize situações que caracterizem o ambiente em que se insere o problema. Neste caso, a utilização de técnicas de simulação e de fundamental importância para a geração de cenários que venham possibilitar a análise de tais situações.

Paralelamente, podemos imaginar um modelo de simulação que

interaja com um sistema especialista nas seguintes condições (Robert O'Keefe [1986]):

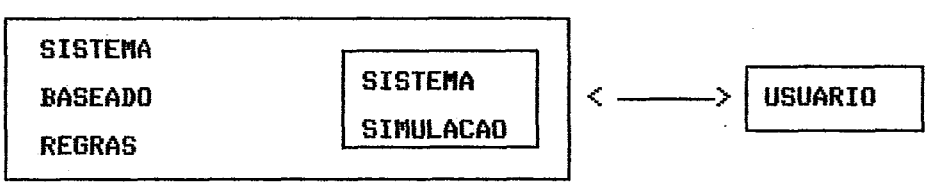
1 - O modelo de simulação interroga um sistema especialista. Por exemplo: um simulador de trens em uma malha ferroviária interroga um sistema especialista que decisão relativa a um conflito de cruzamento deve ser tomada (terceiro caso da figura 2);

2 - Sistemas especialistas podem utilizar resultados de simulações para definir novas regras ou estratégias de controle, o que pode ser considerado como uma forma de aprendizado pela elaboração de novas regras ou unidades de conhecimento denominadas "chunking" (J. E. Laird, A. Newell e P.S. Rosenbloom [1987]), (segundo caso da figura 2) e, finalmente,

3 - Podemos testar um sistema especialista utilizando para tanto um modelo de simulação. Por exemplo: validar o conhecimento de um sistema de gerenciamento de trens através de simulações, sem a incoveniência de experimentos reais. Isto se aplica, principalmente, a validação de sistemas complexos e em tempo real (primeiro caso da figura 2).

Primeiro caso:

Segundo caso:



Terceiro caso:

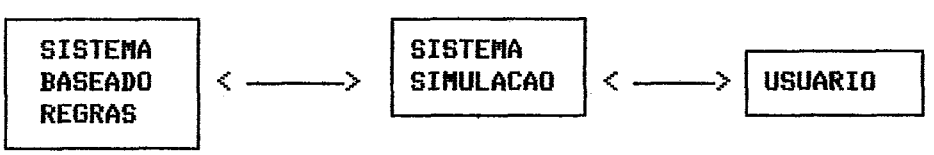


figura 2: integração de sistemas especialistas com sistemas de simulação.

A cooperação simultânea de sistemas especialistas e modelos de

simulação formam sistemas de suporte a decisão mais evoluídos segundo (K. J. Fordyce e G.A. Sullivan [1986]).

Neste sentido, os sistemas DS/DSIM são considerados como um estágio mais avancado da interação entre sistemas especialistas e sistemas de suporte a decisões , combinando técnicas tradicionais dos DSS e ES, com técnicas de análise quantitativa e simulação, interface gráfica e comunicação com janelas. Estes sistemas, não se limitam apenas a responder questões, mas tambem são capazes de formularem alternativas, avaliarem mudancas e predizerem comportamentos, o que é de fundamental importância em sistemas que efetuam a monitoração de planos e cuidam do planejamento estratégico.

Finalmente, devemos destacar o desenvolvimento de ferramentas para a criação destes sistemas, podendo citar como exemplos: " Knowledge-Based Simulation System (KBS)" desenvolvido por (Y. Reddy, M.S. Fox e N. Husain [1985]) e " Rule Oriented Simulation System (ROSS) " desenvolvido por (P. Klahr [1984]).

- "Intelligent Management Systems" (IMS):

Anteriormente, verificamos que um sistema físico simbólico, possui as condições necessárias e suficientes para a tomada de ações inteligentes.

Segundo Mark S. Fox [1989] o espaco do problema operacionaliza o conceito de sistemas físicos simbólicos, através da definição de:

. estruturas de representação simbólicas ou estados;
. operadores ou funções de transformação e
. uma função de avaliação .

Desta forma, o processo de solução de problemas consiste em gerarmos progressivamente estruturas simbólicas a partir da aplicação de operadores até atingirmos uma estrutura equivalente a solução.

Este processo de solução de problemas e conhecido como um processo de busca no espaço do problema, sendo esta busca dirigida por uma função de avaliação e por um conjunto de restrições que caracterizam a região viavél do problema representado por operadores.

Podemos, como exemplo, considerar o metodo "simplex" da programação linear como um processo de busca cuja estrutura de representação é um sistema de equações (vértices do poliedro), os operadores são as operações de fatorização da base e a busca é controlada por uma estratégia de troca de colunas que, incorpora as condições de otimalidade de Kuhn-Tucker que determina o ponto de parada ou a obtenção de uma estrutura solução.

Entretanto, um dos fundamentos básicos da Inteligência Artificial , segundo (Raj Reddy [1988]),é aquele no qual os esforços de busca são compensados pela utilização do conhecimento, e a falta de conhecimento e compensada pelo excesso de busca. Como exemplo, podemos citar uma arquitetura paralela denominada Hitech que joga xadrez em um nivel comparavel a de um mestre sênior. Hitech possui como conhecimento pouco mais de 200 regras, entretanto é capaz de examinar 20 milhoes de posições no tabuleiro em 3 minutos . No outro oposto , temos um mestre sênior de xadrez que raramente explora mais de 200 posições em uma jogada, mas possui um conhecimento estimado em 50.000 padrões ou elementos (H. Simon [1981]).

Paralelamente, podemos identificar no problema de despacho de trens uma situação semelhante, onde o numero de possibilidades de solução de conflitos ou elaboração de um plano de movimento de trens (grade horária) e de magnitude exponencial, apesar de o despachador examinar pouquíssimas alternativas, durante o processo decisório.

Segundo (J.E. Laird, A. Newell e P.S. Rosenbloom [1987]) , que apresentam uma arquitetura denominada SOAR para sistemas inteligentes, descrita , posteriormente, em nosso trabalho, um sistema que se enriquece ou incorpora conhecimento gradativamente, realiza cada vez menos esforço de busca. De um modo geral, esta redução de esforço é compensada por uma forma de controle que expressa a preferência em operadores, reduzindo, desta forma, o fator de ramificação do grafo de estados do problema.

Portanto, estes sistemas realizam uma utilização intensiva do conhecimento, expresso em sistemas de produção, combinando, simultaneamente, métodos de raciocínio, busca e satisfação de restrições na solução de problemas (H. Simon [1983]).

É interessante lembrarmos que os métodos heurísticos de busca utilizados em Inteligência Artificial garantem, em alguns casos, a obtenção da solução ótima do problema, possuindo um forte relacionamento com algumas técnicas e pontos teóricos da Programação Matemática (R. Fonseca Neto e A.A. Pinho [1988]), como por exemplo : dualidade lagrangeana, técnica "branch and bound", métodos de decomposição, condições de otimalidade, formulações, etc.

Entretanto, tratando-se da solução de problemas complexos, mal estruturados, imprecisos, com multi-objetivos, complexidade exponencial, dinâmicos e em tempo real, é mais gratificante a obtenção de soluções satisfatórias em um tempo de computação compatível com a necessidade das ações.

Portanto, podemos situar os sistemas inteligentes no último nível da hierarquia dos sistemas de informação ou apoio a decisões, de forma a propiciar um planejamento eficaz. Estes sistemas, descritos inicialmente por (M.S. Fox [1983]),que retrata o desenvolvimento de um sistema inteligente no Instituto de Robótica da Universidade de Carnegie-Mellon, incorporam basicamente:

1 - Um sistema físico simbólico definidos em termos do espaço do problema, possuindo estados ou estruturas simbólicas, operadores e uma função de avaliação;

2 - Modelos e Teorias de Programação Matemática e Pesquisa Operacional;

3 - Métodos heurísticos de busca em Inteligência Artificial;

4 - Conduta objetiva voltada para a obtenção de uma solução associada a critérios, objetivos e restrições;

5 - Um sistema de produção que abriga o conhecimento de especialistas a respeito do problema, na forma de regras de produção, e que permite a realização de inferências;

6 - Controle da busca através da determinação de preferências nos operadores;

7 - Utilização de técnicas de abstração e decomposição hierárquica (alcance de sub-objetivos);

8 - Capacidade para processamento aproximado e em tempo real;

9 - Representação de conhecimento de longo termo (fatos) na

forma de "frames" incorporando meios automáticos de herança, composição e instanciação;

10 - Interação com modelos de simulação e análise quantitativa de dados;

11 - Uma sofisticada interface com o usuário, incluindo rotinas de visualização gráfica dos resultados e comunicação com janelas , perguntas e respostas interativas, etc. e, finalmente,

12 - Capacidade para tratamento de impasses, atraves da solução de problemas e consequente incorporação de conhecimento, determinando um meio de aprendizado.

Concluindo, esperamos ter apresentado, nesta seção, alguns conceitos básicos, a respeito das principais características de sistemas inteligentes, as quais vao delinear o desenvolvimento de um sistema voltado para o gerenciamento e controle de trens.

2.1.3 - Controle e Gerenciamento Inteligentes:

- Integração de Niveis de Gerenciamento e Controle:

No sentido mais amplo do termo, gerenciamento significa o exercício de pelo menos seis funções fundamentais, incluindo: previsão, planejamento, organização, suporte, direcionamento e controle.

Entretanto, no nosso trabalho, quando nos referimos ao exercício do gerenciamento inteligente, queremos atribuir ao sistema a capacidade de gerenciar planos de ações inteligentes.

O gerenciamento de planos de ações ou planejamento, segundo E. Charniak e D. McDermott [1985] envolve os seguintes níveis ou etapas :

. planejamento tatico ou projeção (elaboração de planos);

. interpretação e execução de planos (tomada de decisões);

. monitoração de planos (controle operacional) e , finalmente,

. replanejamento com reavaliações.

Nas fases ou etapas do planejamento , podemos identificar algumas funções basicas de controle (S. Bennett [1988]). Por exemplo,

a execução de planos (lista de ações sequenciais) pode ser interpretado como a realização de um controle sequencial, a monitoração periódica do efeito do planejamento (tomada de decisões) como a realização de uma forma de controle iterativo, o processo de planejamento tático ou projeção como um tipo de controle supervisor e, finalmente, o processo de replanejamento com reavaliação de planos pode ser interpretado como o "feedback" característico do controle adaptativo.

Sabendo que um sistema para o gerenciamento de trens deve atender questões estratégicas que definem, por exemplo, alterações de "lay-out" e políticas de transporte, e, ao mesmo tempo, se informar de parâmetros operacionais relativos ao posicionamento de trens e detalhes do sistema de sinalização , perguntamos, como conseguir a integração dos níveis de planejamento, envolvendo, também , a tomada de decisões estratégicas e as funções básicas de controle ?

Segundo J.H. Moore e M.G. Chang [1980], os sistemas de suporte a decisão são mais aplicados à tarefas relativas ao planejamento estratégico e tático do que ao controle ou planejamento operacional, não sendo, entretanto,o caso dos sistemas inteligentes de gerenciamento, aos quais competem o exercício de todos os níveis de planejamento, como também, o atendimento a algumas funções estratégicas, bem como, as funções de controle principais.

- Integração de Tecnicas de Inteligencia Artificial, Teoria de Sistemas e Micro-Computação:

Te Xu Yen [1985] propõe uma combinação de técnicas de Inteligência Artificial, Teoria de Sistemas e Micro-Computação como forma de atender aos requisitos e necessidades dos sistemas inteligentes (ver figura 3). Propoe, também, um esquema de decomposição hierárquica e coordenação que permite reduzir a complexidade dos sistemas e definir diferentes níveis de conhecimento e tomada de decisões.

Este esquema de decomposição-coordenação é de fundamental importância no gerenciamento de trens, visto que o controle de tráfego centralizado na ferrovia opera em seções estanques, sendo de grande

necessidade a tarefa de coordenação e supervisão exercida pelos controladores e engenheiros de tráfego.

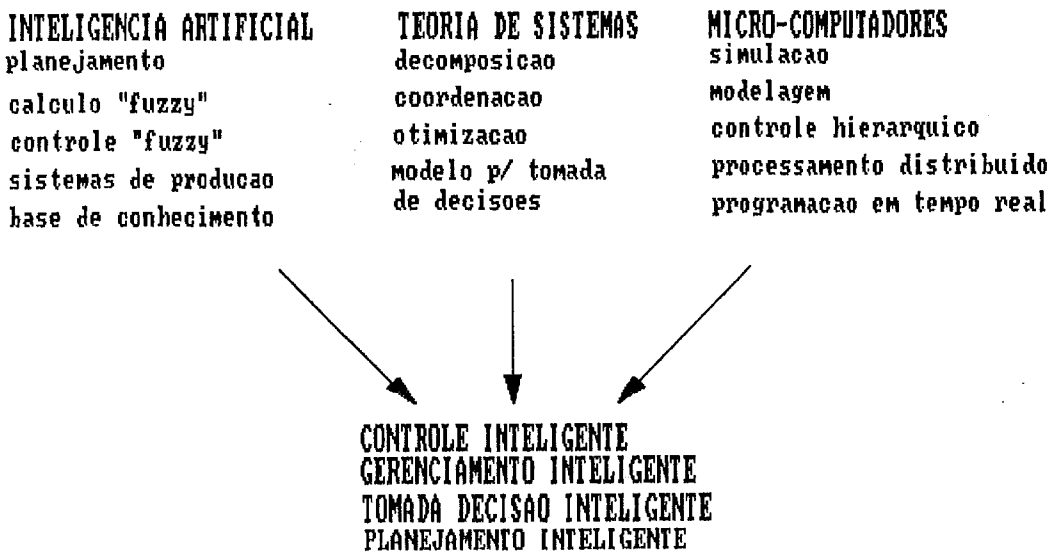


figura 3: integração de técnicas no desenvolvimento de sistemas inteligentes.

Desta forma, principalmente pela modernização dos sistemas de controle de tráfego ferroviário, achamos viável uma integração em tempo real,das funções de gerenciamento com as funções de controle em todos os níveis, formando um sistema inteligente. Neste sistema, o elemento central sera o controlador, que possibilitara a execução do planejamento através do controle de trafego centralizado.

Um exemplo marcante do desenvolvimento de sistemas inteligentes pode ser encontrado no trabalho de (Y. Lirov, E.Y. Rodin, B.G. McElhaney e L.W. Wilbur [1988]) que descreve um modelo de simulação e planejamento de um sistema de controle de combate aéreo baseado na teoria de jogos diferenciais, que permite uma abordagem discreta do problema. O modelo utiliza redes semânticas para a representação do conhecimento e um processo de planejamento baseado na construção de um sistema hierárquico de objetivos dinâmicos, definido para cada centro de

controle (jogador), utilizando para a geração do modelo um algoritmo A* e um sistema de produçãö que orienta a introdução de restrições a cada nível, orientando o processo de busca.

Paul E. Lehner [1986] revisa em seu trabalho a aplicação de técnicas de Inteligência Artificial em funções de comando e controle em problemas de decisões militares. Mais, especificamente, Lehner descreve a utilização de sistemas especialistas como forma de prover uma interface inteligente para o suporte de decisões em tempo real. No trabalho e descrito, tambem, a viabilidade da integração de um sistema de planejamento interativo e hierárquico, SIPE (D.E. Wilkins [1984]), tambem descrito posteriormente, com funções de comando e controle, principalmente, quando se trata da aplicação em problemas.envolvendo o gerenciamento e escalonamento de recursos, como exemplo: o comando e controle de operações em um campo de batalha.

2.2 - Sistema Inteligente para o Gerenciamento do Tráfego Ferroviário (SIGT)

Nesta seção, vamos procurar descrever o sistema inteligente proposto para o gerenciamento do tráfego de trens em uma ferrovia, ver figura 4. Vamos tentar caracterizar este sistema descrevendo seus objetivos principais, sua composição ou arquitetura básica, aspectos da organização hierárquica e, finalmente, sua capacidade para o desempenho de funções básicas desejáveis, incluindo algumas propriedades relevantes.

2.2.1 - Questionário Básico:

Primeiramente, apresentaremos um questionário, que foi de fundamental importância para o desenvolvimento do sistema inteligente para o controle e gerenciamento.

As respostas para o questionário foram levantadas através de entrevistas com pessoas da ferrovia (RFFSA/SR3 - Juiz de Fora), que atuam no controle e gerenciamento de trens, como por exemplo: despachadores, supervisores de tráfego, engenheiros de transporte, gerentes, planejadores, etc., e constituíram uma importante fonte de conhecimento que orientaram os objetivos funcionais, a arquitetura,

enfim, as caracteristicas básicas do sistema SIGT.

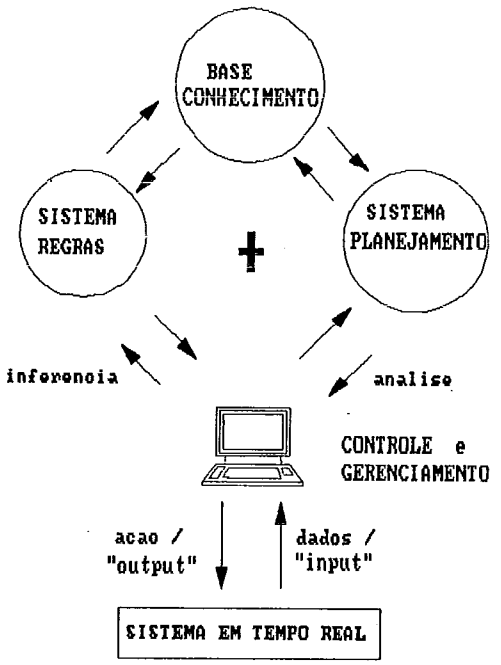


DIAGRAMA DE CONTROLE E GERENCIAMENTO

figura 4: concepção básica do sistema SIGT.

O questionário foi apresentado na forma de um conjunto de perguntas e afirmativas, divididas em grupos, segundo questões de interesse.

1 questão: a respeito dos objetivos do problema :

- Quais são os objetivos principais ?

. minimizar os tempos de atraso (otimalidade do sistema);

. manter cada trem ou grupo de trens em faixas de horário admissíveis, atendendo à níveis de satisfação;

. respeitar regras e prioridades estabelecidas (estratégias superiores, seguranca do sistema);

- manter a estabilidade do sistema (balanceamento do trafego).

2 questão: a respeito da prioridade dos trens :

- Como se comporta a prioridade dos trens ?

. são prioridades estáticas (não variam ao longo do planejamento);

. são prioridades dinâmicas (variam ao longo do planejamento);

. dependem do desempenho de cada trem ou de um grupo de trens (classe);

. dependem do balanceamento do sistema;

. são prioridades altamente dinâmicas (variam ao longo do plano de horários);

3 questão: a respeito das estratégias de solução de conflitos :

- Existem regras de decisão de aplicação geral ou são regras específicas ?

- Observam o comportamento global do sistema ou são decisões localizadas ?

- Atendem a uma política de planejamento tático (plano de rotas) pré-estabelecida ?

- Necessitam de informações a respeito dos objetos (trens) e das entidades (malha física) do sistema ?

- Atendem a uma política de planejamento estratégico (algum procedimento, regra ou prioridade) pré-estabelecida ?

4 questão: a respeito da elaboração do plano de rotas (grade horária):

- Existe um horizonte de planejamento definido (cumprimento das rotas a serem elaboradas) ?

- O plano de rotas elaborado e perseguido ?

- Qual a flexibilidade em torno dos planos elaborados (violação das restrições e objetivos do problema) ?

- Como e quando reelaborar planos ?

- Para se atingir um plano viável, a construção do mesmo e realizada resolvendo-se os conflitos em uma ordem de prioridade natural a partir de um determinado instante (conflito localizado) ? Podemos dizer que esta forma de construção e direcionada segundo a ocorrência de conflitos ?

- Existe algum esquema especial para prevenção de bloqueios (conflitos sem solução) ? Ou seja, este problema e tratado a nível tático de que forma ?

- As regras locais (relativas a objetos e restrições físicas específicas) influem na concepção do plano de rotas ?

5 questão: a respeito do conhecimento necessário a elaboração do sistema de controle e gerenciamento de trens:

- Como são estabelecidas as regras mais gerais de gerenciamento e controle na ferrovia ?

- Como são estabelecidas as regras mais específicas ?

- O conhecimento relativo as regras gerais e específicas se fundem (devem constar em uma mesma estrutura de representação),ou são conhecimentos distintos que não se interagem ?

- O conhecimento relativo as regras gerais e específicas apresentam algum grau de hierarquia ?

- E possível visualizar no sistema um conhecimento extensional (conhecimento descritivo e declarativo do sistema , relacionado a representação e atualização dos dados e atributos dos objetos) e um conhecimento intencional (conhecimento imperativo e procedural do sistema, relacionado a representação de regras e restrições do sistema)?

- O conhecimento traduzido nas regras é, de um modo geral, monotônico ou não monotônico, ou seja, a verdade das regras prevalece sempre ou e dependente de parâmetros do sistema (como hora e local) ?

- As regras estão relacionadas (especificidade) ao comportamento dos objetos (trens, classes de trens, seções) e condicionantes (obstáculos, acidentes, etc.) do sistema ?

- Existe algum conhecimento na forma de padrões (reconhecimento de situações similares) que facilite a identificação de uma linha de ação comum ?

- Existe algum tipo de conhecimento intuitivo que auxilia na tomada de decisões ?

- Este conhecimento, se existente, e fruto de experiência, aprendizado, julgamento ou adquirido de forma espontânea ?

- O conhecimento de como controlar e gerenciar o sistema pode ser formalizado ?

- Como podem ser adquiridas e manipuladas as informações necessarias a operação do sistema em tempo real ?

- Existe algum tipo de informação adicional que pode melhorar o controle e gerenciamento do sistema ?

6 questão: a respeito da imprecisão, incerteza e aleatoriedade das variáveis, conhecimento e eventos característicos do sistema :

- Como se realizam as previsões dos tempos de percurso dos trens ?

- É possível a utilização de alguma distribuição de probabilidade, tabela de frequências ou distribuição de possibilidade para representar a aleatoriedade e/ou imprecisão dos parâmetros relativos aos tempos de percurso dos trens ?

- As distorções em torno da previsão dos tempos de percurso afetam a otimalidade/viabilidade dos planos de rotas ?

- Existem expressões vagas na composição de alguma regra ?

- A imprecisão e incerteza presentes no sistema, podem ser manipulados, sem prejuízo da consistência do conhecimento ?

- É possível o gerenciamento e controle de trens com ausência de informações ou ignorância em relação a alguns eventos externos ?

- Algumas informações imprecisas, como exemplo: a posição relativa dos trens no painel de controle, se fossem obtidas com absoluta precisão, influenciariam em que grau a performance e qualidade do trabalho do controlador ?

7 questão: a respeito do caráter multi-objetivo, multi-critério e hierárquico do sistema de controle e gerenciamento de trens:

- Existe uma hierarquia bem definida no sistema decisão, ou seja, existem objetivos mais gerais (globais) que envolvem metas e estratégias e objetivos mais específicos (locais) relacionados à características físicas e estruturais do sistema ? Podemos citar como exemplo, a criação de plano de rotas seguindo um objetivo mais amplo e a solução de um conflito local com aplicação de uma regra seguindo uma restrição de caráter físico.

- Como é estabelecida a interrelação entre os sistemas de controle e gerenciamento locais e os sistemas mais gerais ? Existe a coordenação característica de sistemas hierárquicos ?

- Os objetivos relativos ao gerenciamento e controle do sistema podem ser expressos através de uma função de avaliação ? Existe a

possibilidade de uma solução ótima (majorante) ?

- Os objetivos relativos ao gerenciamento e controle do sistema são expressos mais eficientemente através de uma formulação flexível, que forneca um tratamento similar as restrições e objetivos do problema? É mais natural que uma solução seja viável e eficiente, do que, necessariamente, ótima ?

- Na hipótese da construção de uma árvore de busca existe a possibilidade de elaboração de um procedimento heurístico que torne mais eficiente o processo de busca (restringindo o espaço de estados ou o conjunto de soluções viáveis do problema) ?

- Existe o caráter compensatório na reavaliação dos objetivos do problema ?

- Existe uma descentralização do sistema de controle e gerenciamento ? Ou seja, existem seções de controle (setores) estanques ? Como é realizado o gerenciamento das interfaces à nível de troca de informações e garantia de otimalidade do sistema ?

- Os objetivos e critérios do sistema de suporte a decisões tem graus de importância relativos ? Por exemplo: muito importante, pouco importante, etc.

- Tratando-se de um sistema dinâmico e em tempo real, existe alterações no grau de importância dos objetivos e critérios ? Por exemplo: a mudanca de prioridade dos trens.

- Estas alterações podem tornar-se internas ao sistema ou necessitam do auxílio do controlador como se verificam nos sistemas interativos ?

8 questão: A respeito da viabilidade de implantação do sistema de controle e gerenciamento de trens:

- Um conhecimento maior sobre variáveis do sistema, como regras de sinalização, atributo de objetos, prioridade dinâmica dos trens, plano de rotas, permitiria um melhor controle e gerenciamento do mesmo ?

- A existência de um processo de coordenação entre os vários setores (seções de controle) seria importante para a otimalidade e satisfação dos objetivos globais (viabilidade) do sistema ?

- A possibilidade de se examinar um número maior de soluções viáveis na construção de um plano de rotas melhoraria o desempenho da circulação dos trens ?

- A utilização de um sistema de controle e gerenciamento

dinâmico e flexível é superior a utilização de um sistema de suporte as decisões estático e sem flexibilidade ?

- A reelaboração de plano de rotas, sendo um trabalho tedioso e repetitivo, e propício a automação ?

- A formalização do conhecimento adquirido pelos controladores pode ser útil em programas de treinamento ? Com a integração de bases de conhecimento e técnicas de simulação é possível ampliarmos esta base de conhecimento e tornarmos mais eficiente o sistema de suporte as decisoes ?

2.2.2 - Caracterização do Sistema:

- Funções e Objetivos Básicos:

O sistema SIGT compreende um número abrangente de funções necessárias à realização do planejamento relativo ao gerenciamento e controle de trens. Considerando que a malha ferroviária foi devidamente decomposta em setores com respectivas interfaces e sujeitas a um controle de tráfego setorizado, foram propostos três grupos de funções principais, seguindo as etapas ou níveis de planejamento apresentados anteriormente, gerando uma estrutura hierárquica, ver figura 5.

1- Nível estratégico com domínio centralizado:

. Coordenar interfaces dos setores quanto a tomada de decisões;

. Coordenar a realização do planejamento tático (elaboração de planos adjacentes);

. Manter uma base de conhecimento e um sistema de produção de caráter geral;

. Determinar e avaliar objetivos e metas, e

. Estabelecer estratégias e prioridades estáticas.

2- Nível tático com domínio setorizado:

. Realizar o planejamento tático (elaboração do plano de rotas ou grade horária);

. Coordenar o planejamento temporal (interface entre planos consecutivos);

. Manter um sistema de regras e uma base de conhecimento com caráter particular;

. Reavaliar prioridades dinâmicas e objetivos, e

. Realizar o replanejamento (reconstrução de planos).

3 - Nível operacional com domínio localizado:

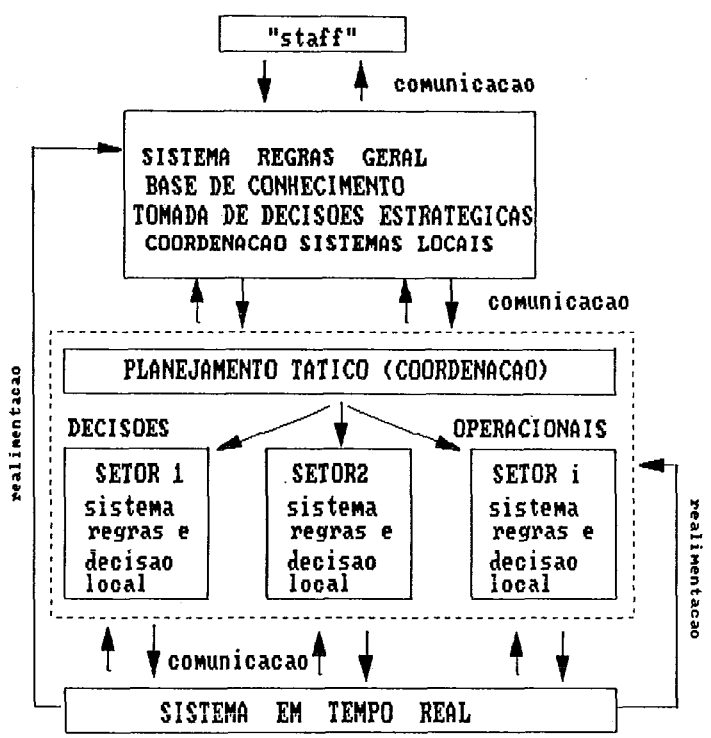
. Tomar decisões localizadas (solução de conflitos de cruzamento e ultrapassagem, etc) segundo o plano de ações;

. Monitorar o planejamento quanto a viabilidade das ações (satisfação das restrições e objetivos);

. Garantir a segurança no movimento de trens;

. Determinar a realização do replanejamento, e

. Receber, consultar e atualizar informações da base de conhecimento, como exemplo: posicionamento de trens, tempos de percurso, "status" dos pátios de cruzamento, etc.



TOMADA DE DECISAO INTELIGENTE
(SISTEMA DE CONHECIMENTO HIERARQUICO)

figura 5: estrutura hierárquica do sistema SIGT.

George Chryssolouris [1989] propõe um modelo hierárquico semelhante ao SIGT para o desenvolvimento de um sistema inteligente aplicado a produção de manufaturados. Chryssolouris identifica três níveis de decisões associado as facilidades de um centro de produção e respectivos níveis estruturais. Desta forma, ele associa a decisões estratégicas as funções de empreendimento associadas ao estabelecimento de metas e estratégias pelos gerentes, como exemplo uma faixa de produção ou lucratividade anual. A um nivel tático de planejamento as decisões estão associadas ao gerenciamento da produção e voltadas para a distribuição de recursos e estabelecimento de ordens de trabalho. Finalmente, ele destaca as decisões operacionais associadas aos centros de trabalho e relacionadas aos processos de produção estabelecendo as tarefas básicas.

Se fizermos uma analogia da hierarquia proposta por Chryssolouris com o sistema ferroviário, podemos admitir que no nível estratégico são decididas metas de transporte e estabelecido prioridades de circulação. A um nivel tático de planejamento são elaborados os planos de rotas ou movimento, obedecendo as metas sugeridas (níveis de atraso) bem como as prioridades de circulação, para cada setor da ferrovia e finalmente, a um nivel operacional , cada setor recebe seu respectivo plano, efetuando as ações ou decisões operacionais básicas, possibilitando, desta forma, a movimentação de trens, solução de conflitos, manutenção da via, etc.

Dentre os objetivos principais do SIGT podemos destacar:

1- Otimizar o despacho de trens através da elaboração de planos de rotas ou tabelas de horários;

2- Automatizar, parcialmente, este despacho através da monitoração do planejamento em tempo real e, finalmente,

3- Assegurar a segurança do movimento e a viabilidade do planejamento através da reelaboração de planos com a realização do replanejamento.

De forma resumida, podemos considerar o desempenho de três funções básicas pelo sistema, de modo a alcançar os objetivos

estabelecidos, como mostra a figura 6.

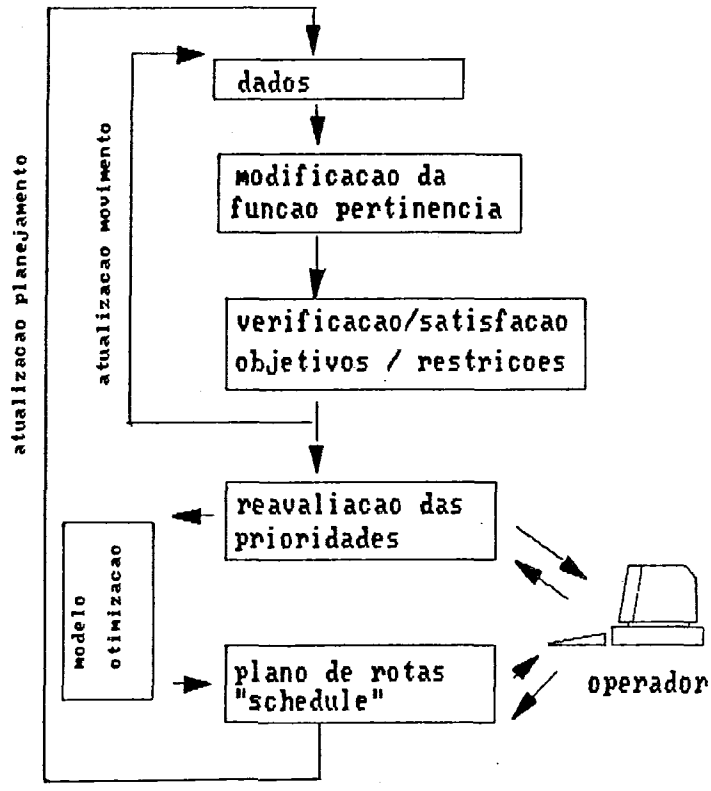


figura 6: planejamento, monitoração e replanejamento.

Queremos ressaltar, que a automação proposta e parcial pois o sistema de gerenciamento funcionará em laço aberto com o sistema de controle centralizado, oferecendo ao despachador sugestões para a realização do despacho de trens. Desta forma,a tomada de decisão final e responsabilidade, serão sempres do despachador, garantindo a segurança do controle de trens. Também, como pode ser observado na figura 6, na operação em tempo real, as funções de planejamento e replanejamento se confundem.

Estes são considerados os objetivos principais do SIGT quanto a realização do gerenciamento e controle, entretanto, o mesmo possui objetivos e funções secundárias relacionados a possibilidade de estudos, análise e avaliações pelo uso integrado de técnicas de Otimização, Simulação e Inteligencia Artificial como veremos em capítulos seguintes.

2.2.3 - Arquitetura Básica do Sistema:

Seguindo a descrição de sistemas inteligentes, podemos identificar no SIGT três componentes principais:

- um sistema físico simbólico associado a um modelo matemático;
- um sistema de produção e
- uma base de conhecimento.

Existem, tambem, outros componentes de apoio, necessários ao desempenho das funções de planejamento, podendo citar:

- um modelo de simulação orientado pela solução de conflitos;
- um modelo de monitoração e
- uma interface inteligente contendo janelas de comunicação e visualização gráfica de resultados.

Os componentes do sistema serão analizados com bastante profundidade nos capitulos seguintes de nosso trabalho. Entretanto, a seguir, teceremos alguns detalhes descritivos , fundamentais a uma primeira idealização do modelo SIGT. Desta forma, possibilitamos ao leitor, um entendimento inicial a respeito do funcionamento do sistema.

- Sistema Físico Simbólico Associado a um Modelo Matemático:

Este sistema é responsável pela realização do planejamento tático ou elaboração da tabela horária (grade de trens).

O modelo matemático associado ao problema de despacho de trens constitui um modelo de programação multi-objetiva "fuzzy", contendo as restrições estruturais do problema de "scheduling" (restrições de sequenciamento e restrições de precedência disjuntivas) e um conjunto de restrições flexivéis que representam funções de pertinência e avaliam o grau de satisfação dos objetivos. A solução do problema segundo (J.J. Buckley [1983]) garantirá a geração de pontos eficientes ou pontos não dominados.

O sistema físico simbólico possui como estruturas representativas ou estados, tabelas de tempos que representam o instante

de ocupação de cada trem em cada seção do setor ferroviário sob planejamento. Estas tabelas de horários possibilitarão a formação de grades de trens ou diagramas espaço x tempo , com as respectivas soluções de conflitos.

A geração e mudanca sucessiva de estados será realizada pela resolução de conflitos na ordem cronológica, determinando uma árvore binária de busca. O processo de avaliação de estados será feito pela utilização de uma heurística admissível que avaliará o nível de satisfação dos objetivos em função do instante de ocupação de cada trem, na ultima seção de suas respectivas rotas. Desta forma, a cada trem ou grupo de trens, estará associado uma função de pertinência linear por partes, côncava, relativa a um critério que estabelece um limite de atraso aceitável, determinando uma hierarquia de objetivos flexível amarrados a um conjunto de prioridades.

A agregação das funções de pertinência deve obedecer ao operador sugerido por (R.E. Bellman e L.A. Zadeh [1970]) produzindo uma função de avaliação única , monotonicamente decrescente , que garantirá a obtenção de soluções otimas .

Para a realização da busca heurística serão utilizados dois métodos conhecidos, respectivamente, como A* (P.E. Hart, N.J. Nilsson e B. Raphael [1968]), IDA* (R.E. Korf [1985]) e um terceiro método que realiza sucessivas buscas em profundidade.

- Sistema de Produção:

O sistema de produção desenvolvido para o SIGT, incorpora, aproximadamente, 150 regras e metaregras, organizadas hierarquicamente na forma da figura 7.

No nível inferior da hierarquia, verificamos a condição de algum trem parar em um pátio de cruzamento (considerando a capacidade do pátio, status, demarragem, heurísticas, etc.) em caso de conflito em linha singela; ou avançar no casos de conflito em linha duplicada, determinando os seguintes sub-objetivos: obrigatória, permitida e não permitida.

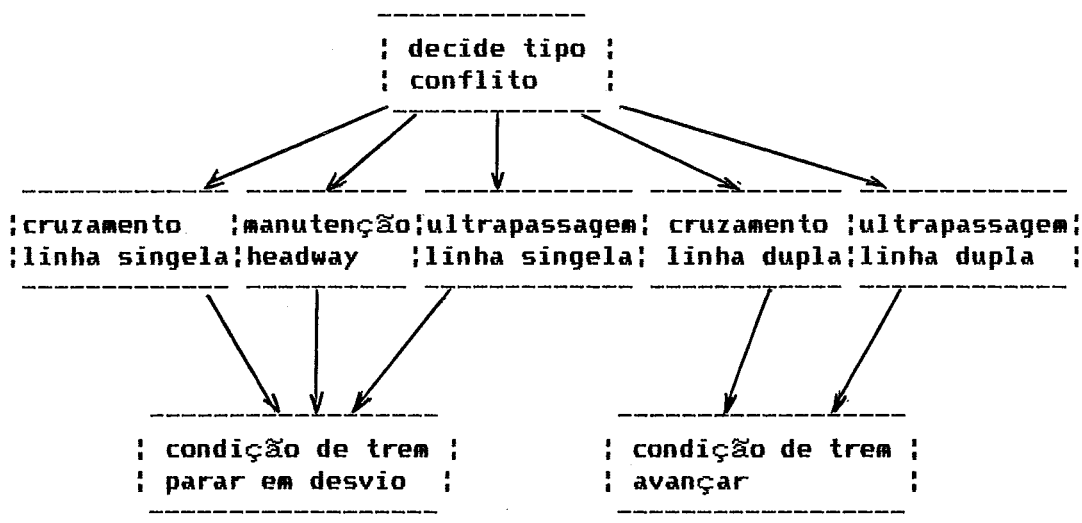


figura 7: hierarquia dos módulos de regras.

No nível médio da hierarquia, será determinada a preferência em algum operador segundo o tipo de conflito e as condicionantes do sistema. Como exemplo: preferência para trem impar ou par parar em cruzamento, etc.

Finalmente, no nível mais alto, temos um único módulo que determina o tipo de conflito e invoca o módulo de regras correspondente para a determinação da preferência.

Os objetivos principais do sistema de produção são:

1- Garantir a viabilidade de restrições de capacidade (capacidade de pátios de cruzamento em número de linhas) e restrições operacionais (capacidade de pátios quanto ao tamanho de trens, etc.);

2- Prevenir a ocorrência de bloqueios ou impasses (conflitos sem solução);

3- Possibilitar uma busca inteligente através da utilização de heurísticas, como exemplo: dar preferência para a formação de onda de trens e, finalmente,

4 - Reduzir o espaço de estados do problema através da determinação de preferência em operadores.

A arquitetura do sistema de produção incorpora , basicamente:

1 - Um conjunto de produções constituidos de regras e metaregras;

2 - Uma máquina de inferência que é, essencialmente, um interpretador de regras com raciocínio para frente;

3 - Estratégias de controle que decidem qual a melhor regra a ser executada;

4 - Uma memória de trabalho constituída por elementos na forma objeto-atributos-valor, ou seja, a cada objeto é permitido a existência de múltiplos atributos mapeados em um conjunto de valores legais e

5 - Um mecanismo de explanação de raciocínio explicando a cada consulta O que?, Porquê ? e Como ? foi determinada alguma das situações de preferência (ver figuras 8 e 9) .

- Base de Conhecimento:

A base de conhecimento desenvolvida para o SIGT é apoiada na utilização de "frames" (P. Fiker e T. Kehler [1985]) ou estruturas de representação de conhecimento que permitem a possibilidade de herança, composição e instanciação na manipulação de objetos.

Para o sistema SIGT foi desenvolvida uma linguagem para a construção de frames baseada na idéia de abstração de classes, permitindo, a exemplo de (G. Brewka [1987]), uma hierarquia de classes e instâncias com metaclasses, ou seja , classes cujas instâncias (variáveis de classes) são consideradas classes. Desta forma, concordando com o modelo orientado para objetos de (J-P Briot e P. Cointe [1987]) existem na nossa linguagem apenas dois tipos de objetos: classes e instâncias terminais (não classes).

Serão definidas três "frames" principais :

A primeira representando o conhecimento relativo a classe de trens. As principais informações ou "slots" desta estrutura serão:

- tempos de percurso;
- peso, potência e comprimento, e
- rota e prioridade estática.

```
explanacao de raciocinio (S)im ou (N)ao ?

                                                    O Que
                                                    Por Que
nao ha preferencia                                  Como
secao >  3 trem1 >  1 trem2 >  4                    exit


(determine preferencia para parar)  regra (  1)
(nao ha preferencia)  regra ( 46)
(determine condicao de trem2 parar na secao)  regra ( 11)
(condicao e permitida)  regra ( 79)
(determine condicao de trem1 parar na secao)  regra ( 10)
(condicao e permitida)  regra ( 78)
(conflito e do tipo cruzamento)  regra ( 37)
```

figura 8: explanação de raciocínio O quê? PorQuê?

```
  (condicao de parar trem1 e permitida)
E (condicao de parar trem2 e permitida)
E (prioridade do trem1 e igual prioridade do trem2)  regra ( 46)

                                                    O Que
  (condicao de parar trem2 nao e conhecida)         Por Que
E (trem2 nao aguarda em patio de cruzamento)  regra ( 11)  Como
                                                    exit
  (sentido do trem e igual a par)
E (e possivel demarragem no patio posterior)
E (comprimento do trem e menor que o patio posterior)
E (patio posterior esta desocupado)
E (trem nao aguarda em cruzamento)  regra ( 79)

  (condicao de parar trem1 nao e conhecida)
E (trem1 nao aguarda em patio de cruzamento)  regra ( 10)

  (sentido do trem e igual a impar)
E (e possivel demarragem no patio anterior)
E (comprimento do trem e menor que o patio anterior)
E (patio anterior esta desocupado)
E (trem nao aguarda em cruzamento)  regra ( 78)

  (sentido do trem1 e igual a impar)
E (sentido do trem2 e igual a par)  regra ( 37)
```

figura 9: explanação de raciocínio Como?

E interessante observarmos que as informações que descrevem uma classe de trens são inferidas por todos os trens que pertencem a esta classe ou herdam as características da mesma. Podemos desta forma, criar objetos (instâncias de "frames") relativos a trens de minério, trens de carvão, etc.

A segunda representando o conhecimento relativo a trens. As principais informações ou "slots" desta estrutura serão:

- hora de entrada;
- código e
- instância da "frame" ou classe ao qual pertence.

Neste caso, as informações existentes especializam determinado trem associado a uma repectiva classe (herança). Podemos , desta forma, criar objetos (instâncias terminais) relativos a trens pertencentes a alguma classe previamente definida, herdando as informações comuns.

E a terceira representando o conhecimento relativo a seções (trecho da ferrovia). As principais informações ou "slots" desta estrutura serão:

- número de linhas (linha dupla ou singela);
- número de blocos, comprimento dos blocos;
- tempos de "headway";
- status (ocupada, desativada, livre , etc.) e
- pátios de cruzamento (tamanho, status, perfil).

Podemos,tambem, criar objetos (instâncias terminais) relativos as seções da ferrovia.

Convem ressaltar, que as estruturas de representação do conhecimento são implementadas como tipo abstrato de dados e utilizam a memória dinâmica (memória de "heap") definida pela linguagem de programação.

É possível a manipulação do conhecimento armazenado nas "frames" através da consulta ou modificação de seus "slots" (ver figura 10).

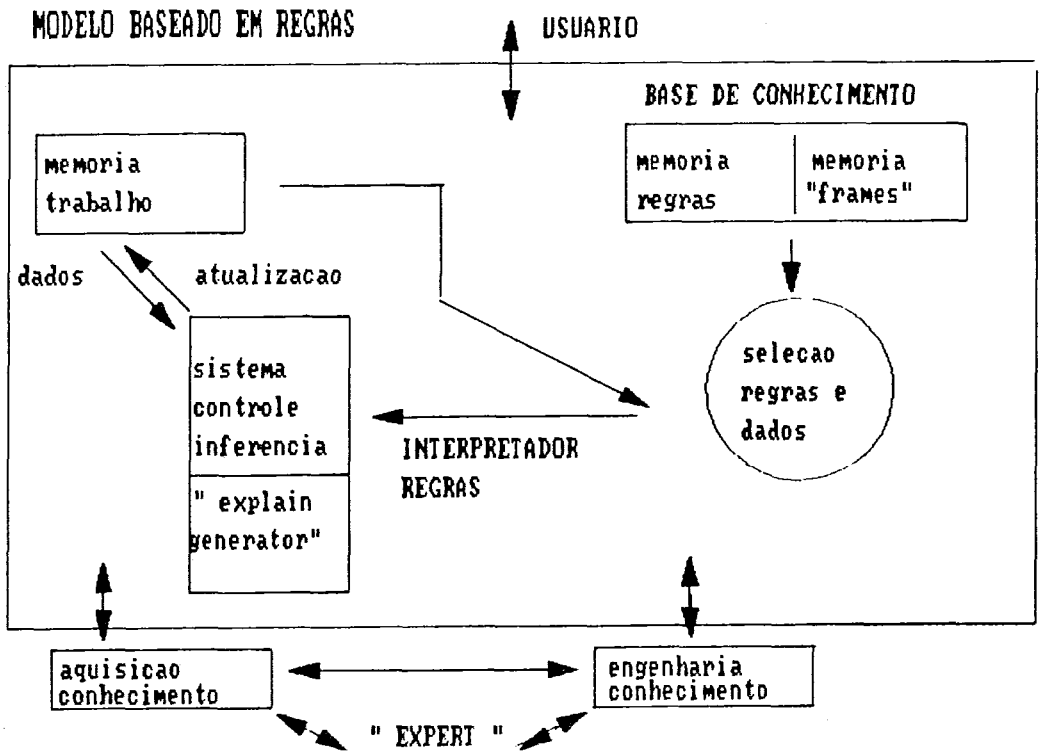


figura 11: arquitetura combinando "frames" e regras

2.2.4 - Capacidade e Funcionalidade do Sistema SIGT

- Modelagem:

O problema de despacho de trens foi modelado matematicamente pelo SIGT como um problema de "job-shop scheduling" . Esta formulação foi inicialmente proposta por B. Szpigel [1972], sendo que, no modelo de Szpigel, voltado para o planejamento tático, o objetivo do problema retrata a otimização global do sistema (minimiza os tempos de atraso).

Entretanto, a um nível operacional, o problema de despacho de trens caracteriza-se como um problema dinâmico, impreciso e mal estruturado (não existe tratamento matemático adequado que garanta a obtenção de soluções ótimas), existindo vários objetivos e critérios que possam influenciar a escolha de uma boa solução. Portanto, para que fosse possível resolvermos de forma adequada o problema, utilizamos uma modelagem multi-objetiva "fuzzy" para o mesmo, dando um tratamento similar aos objetivos e restrições (C.V. Negoita [1981]). Esta

- Combinando "Frames" e Regras :

No SIGT, utilizamos muitas vêzes nas premissas de regras, valores relativos a "slots" de "frames". O valor dos "slots", que representam informações atributivas ou relacionais de determinados conceitos, e considerado um conhecimento na forma declarativa ou segundo P. Harmon e D. King [1985], na forma de pares atributos-valores (AV). Entretanto, estes valores podem representar um conhecimento na forma procedural (procedimentos ou metodos e coleções de regras de produções associados aos "slots") ou um tipo de meta-informação (referência a classes).

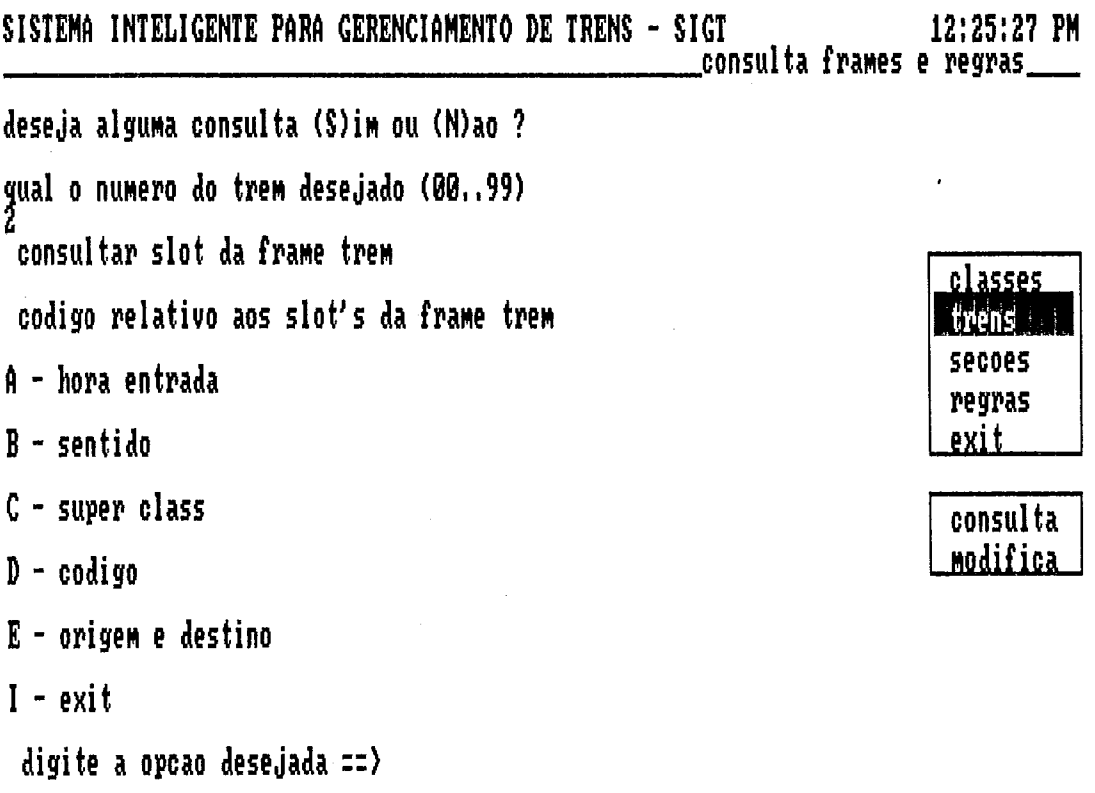


figura 10: manipulação de "frames".

Desta forma, sugerimos no SIGT o desenvolvimento de um modelo de regras, que utiliza o conhecimento representado nas "frames", originando a seguinte arquitetura básica (ver figura 11).

formulação na forma de inequações, que permite a obtenção de soluções satisfatórias, é muito mais eficaz que o processo de otimização convencional.

Quanto a geração de um modelo para o sistema físico simbólico, sugerimos a construção da representação inicial do estado raiz, considerando apenas as restrições de sequenciamento do modelo matemático. Em seguida, em cada nível do processo de busca, adicionamos uma restrição de precedência (caracterizando a aplicação de um operador) no sistema de equações representado pela tabela de tempos. Entretanto, o tamanho da estrutura inicial (dimensões da tabela de tempos) sera, sempre preservado, admitindo-se uma troca de restrições no sistema de equações,. como veremos mais adiante.

Portanto, no nosso problema, o conhecimento necessário a realização da busca (solução do problema) esta modelado como um conhecimento procedural (na forma de restrições e objetivos) representado por uma tabela de tempos , um conjunto de operadores e um conjunto de objetivos (funções de pertinencia); um conhecimento operacional (na forma de regras) representado por um sistema de produção e um conhecimento declarativo representado pelas "frames" ou estruturas de informação.

- Otimização:

Na modelagem inicial do problema de despacho de trens (B. Szpigel [1972]) é utilizado uma técnica de programação linear relaxada seguido de uma busca do tipo "branch e bound" para se determinar a solução otima que minimiza os tempos de atraso do sistema.

Na nova formulação, atraves do uso do algoritmo A*, obtemos uma solução ótima para a função de avaliação do problema, quando o mesmo e equacionado como um problema de Max-Min. Na verdade, determinamos, segundo L.F.B. Baptistella e A. Ollero [1980], uma solução com mais alto grau de pertinência (maximizante) de um conjunto de soluções não-dominadas.

O conjunto de soluções não-dominadas, segundo J.J. Buckley [1983], são todos os possíveis pontos viáveis, correspondentes a

agregação dos objetivos (funções de pertinência) pelo uso do operador "min" (representando a interseção de objetivos), caso as funções sejam monotonicamente decrescentes, lineares ou lineares por partes.

Podemos tambem, considerar a resolução do problema de despacho de trens na forma de inequações, especificando um nível mínimo de satisfação admissível (limite) para a função de avaliação.

- Planejamento e Replanejamento:

A capacidade de elaboração e reconstrução de planos é, sem dúvida alguma, a mais importante e desejável característica de sistemas inteligentes. Todo planejamento ou atividade de elaboração de planos , obedece a alguns princípios e critérios básicos que incluem a racionalidade, flexibilidade e satisfação.

O emprego da racionalidade durante o processo de desenvolvimento de um plano visa minimizar a utilização de recursos e atingir objetivos e metas específicos.

A flexibilidade garante a realização do planejamento/ replanejamento dentro do período desejável atendendo as restrições de tempo, características do controle e gerenciamento em tempo real.

E, finalmente, o critério de satisfação garante a obtenção de planos razoáveis orientados por uma estratégia de busca que realize menos esforço computacional, não necessitando, necessariamente, alcançar a solução ótima.

Willian Swartourt [1988] apresenta um artigo refletindo as principais questões sobre planejamento discutidos em uma reuniao de trabalhos promovida pela Agência de Defesa Americana (DARPA). Dentre as questões analisadas destaca-se aquela que aborda a realização do planejamento com incerteza, ausência ou imprecisão de informações e conhecimento.

No sistema ferroviario, temos uma grande imprecisão nos parâmetros que especificam o tempo de percurso dos trens, a hora de entrada em circulação, etc. Ademais, no controle de tráfego centralizado, com rastreamento discreto, não temos o posicionamento exato dos trens na malha ferroviária.

A analise da imprecisão destes parametros é muito importante para o exercício de um planejamento bem sucedido, já que a existência de imprecisão, pode comprometer a viabilidade dos planos de ações.

O sistema SIGT têm a capacidade de realizar o planejamento de forma racional, flexível e eficiente. Utilizando a formulação Max-Min e a estratégia de solução de conflitos na ordem cronológica, é possível a elaboração de planos parcialmente viáveis, a qualquer hora, atendendo as restrições de tempo impostas pela aplicação em tempo real.

Podemos, tambem, utilizar a formulação em forma de inequações e realizarmos buscas em profundidade progressivas fazendo o "backtracking" ao estado de melhor valor. Neste caso, teremos sempre em mão um plano viável, e a busca, terá continuidade, até atendermos ao limite estabelecido para a função de avaliação do problema.

Na etapa de replanejamento, o sistema realiza a reavaliação de prioridades dos trens e reconstrói os objetivos alterando os intervalos de tempos (atrasos admissíveis) que servem de suporte para o desenvolvimento das funções de pertinência. Desta forma, aumentamos o atraso admissível para os trens com boa performance e diminuimos o atraso admissível para os trens com performance ruim.

- Simulação:

O uso de modelos de simulação proporcionam:

1 - Predizer o comportamento de sistemas reais;

2 - Possibilitar a realização de experimentos em um sistema artificial, ao invés de se modificar o sistema real e, finalmente,

3 - Ajudar na escolha de variáveis e parâmetros de decisão relacionadas ao sistema simulado.

Em um sistema ferroviario a simulação permite que todos os componentes e características de uma ferrovia, sejam representados no computador e a mesma ser operada de forma artificial. Todos os parâmetros e variáveis fornecidos ao modelo podem ser alterados de forma diversas, possibilitando a escolha de alternativas e estratégias ótimas , de acordo com as metas e objetivos da ferrovia.

Na concepção de (A.A. Assad [1983]) o modelo de simulação incorporado ao SIGT caracteriza-se como um modelo de simulação de linha. Entretanto, o mesmo foi desenvolvido dentro de uma concepção mais avançada de simulação (R.E Shannon, R. Mayer e H.H. Adelsberg [1985]), diferenciando-se dos modelos de simulação de linha tradicionalmente mais utilizados, como por exemplo o "Route Capacity Model" desenvolvido pela Canadian National Rail em 1981, e o " Train Dispatching and Simulation Model " desenvolvido pela Peat, Marwick Mitcheell and Co. em 1975.

Entre as diferenças principais podemos destacar:

1 - No SIGT a modelagem é orientada pela construção de objetos e o conhecimento e modelado simbólicamente pelo uso de "frames", regras, estruturas simbólicas, etc. Nos modelos tradicionais a modelagem é orientada segundo a construção de diagramas lógicos, contendo fluxos e transações, e o conhecimento é, essencialmente, numérico. O desenvolvimento de um modelo para a simulação de sistemas biológicos utilizando programação orientada para objetos (linguagen Flavors) foi apresentado por (T.S. Larkin, R.I. Carruthers e R.S. Soper [1988]);

2 - No SIGT o mecanismo de avanço do tempo é orientado pela solução de conflitos, permitindo a execução paralela de eventos que não tenham dependência lógica (D. Kumar [1986]). Portanto, cada trem e considerado um processo independente que utiliza recursos da ferrovia (pátio de cruzamento, linha férrea , etc.). A solução de conflitos é equivalente a sincronização dos processos na visão orientada por processos de simulação (W.R. Franta [1977]). Nos modelos tradicionais o mecanismo de avanço do tempo é orientado pela ocorrência de eventos (partida de um trem, chegada de outro trem, etc) ordenados cronológicamente numa lista de eventos;

3 - No SIGT existe uma utilização intensiva do conhecimento, e uma forma de controle da simulação não explícita (heurística). Existe uma cooperação entre o modelo de simulação e um sistema especialista , sendo possível a simulação das produções. Tambem, alternativamente, é possível orientar os passos da simulação quanto a viabilidade das decisões, observando a determinação de preferências. Os sistemas tradicionais não possuem interação com sistemas especialistas, e o

controle da simulação e na forma explícita (algorítmica);

4- Existe no SIGT uma interface amigável e interação contínua com o usuário, possibilitando, através do uso de janelas com opções ou menus:

- Visualizar e modificar progressivamente "slots" das "frames" (estruturas de representação do conhecimento), ver figura 12.

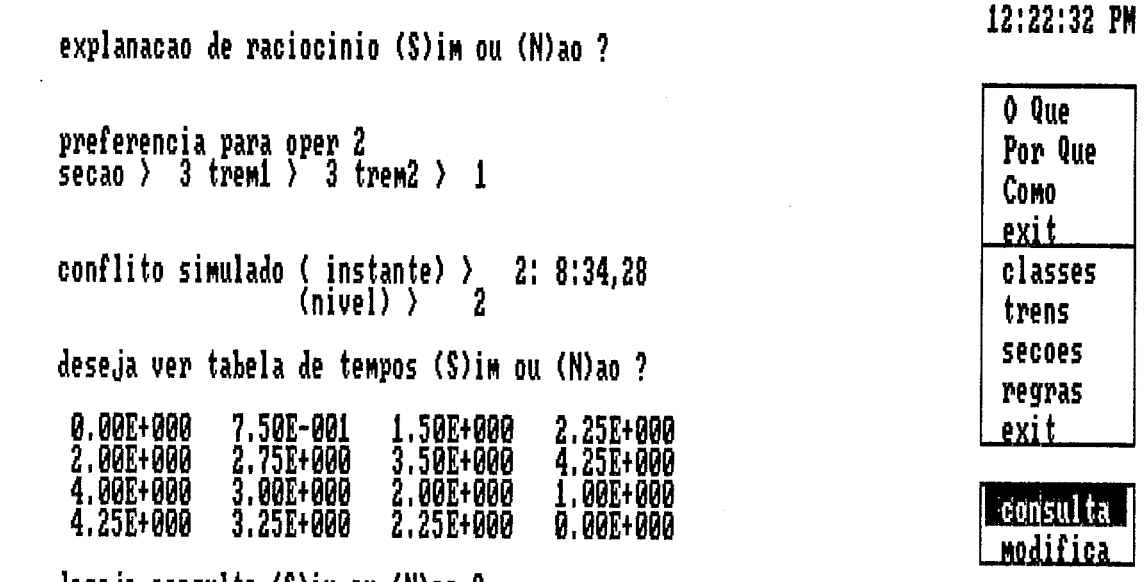


figura 12: manipulação de "frames" e visualização dos estados (tabela de tempos).

- Monitorar o progresso da simulação através da visualização gráfica (grade de trens) e numérica (funções de pertinência e atrasos acumulados) dos resultados parciais, ver figura 13;

- Escolher o operador mais adequado, atraves da interpretação da ação correspondente que é mostrada textualmente em janelas, ver figura 14;

- Consultar o sistemas de produções que informara qual a preferência determinada ? por quê ? e como ?, ver figura 12;

- Retroceder o processo de simulação a qualquer ponto desejado (instantes de ocorrência de conflitos). Desta forma, é permitido ao

usuário tomar uma decisão, avaliar o seu efeito e, em seguida, retroceder ao instante do conflito e tomar a decisão alternativa , permitindo a comparação com a decisão anterior;

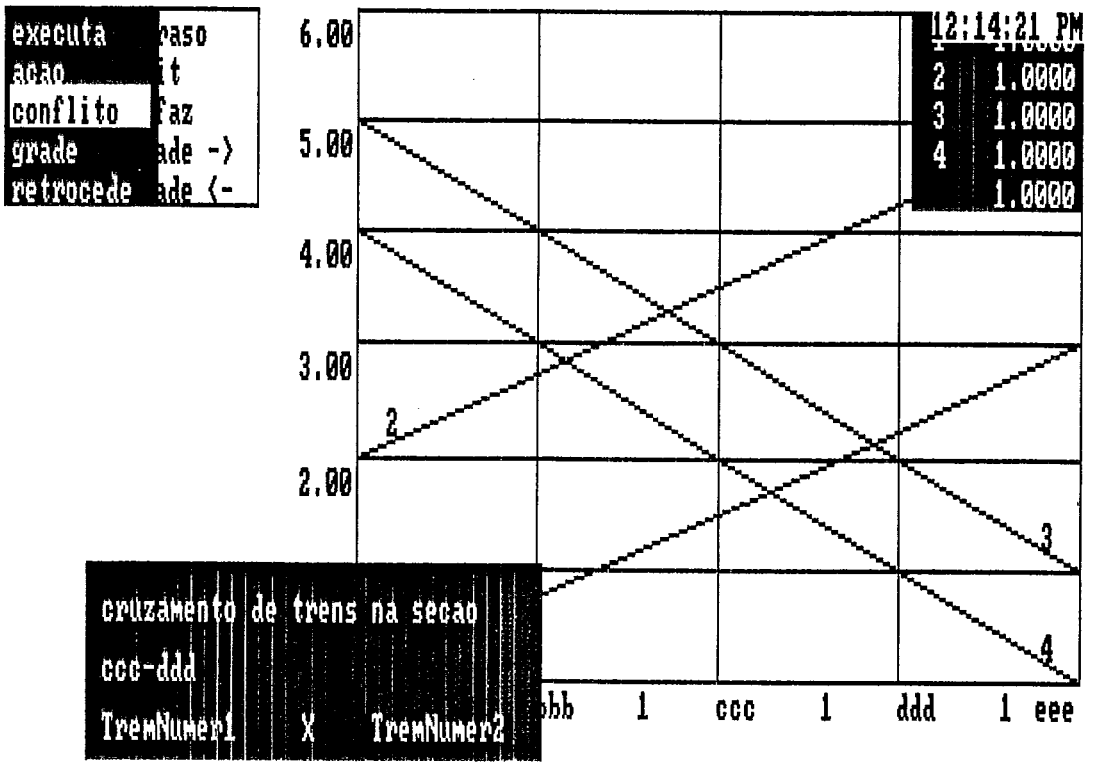


figura 13: monitoração da simulação.

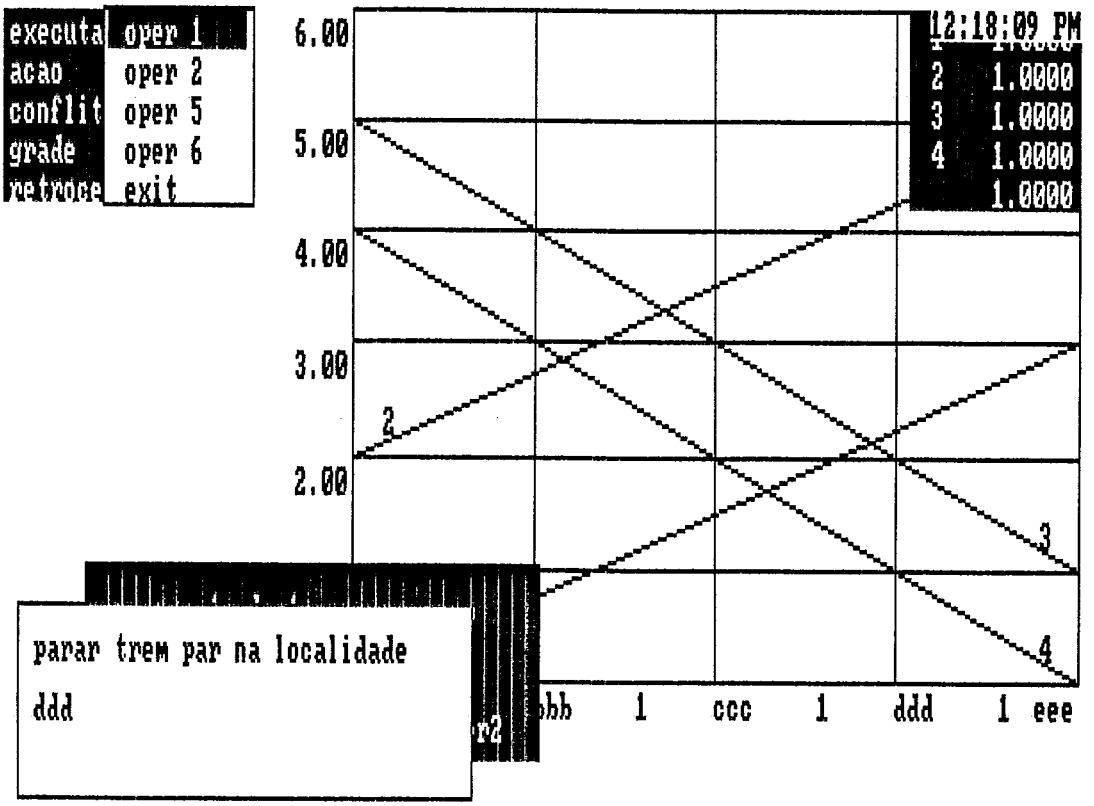


figura 14: escolha de operadores.

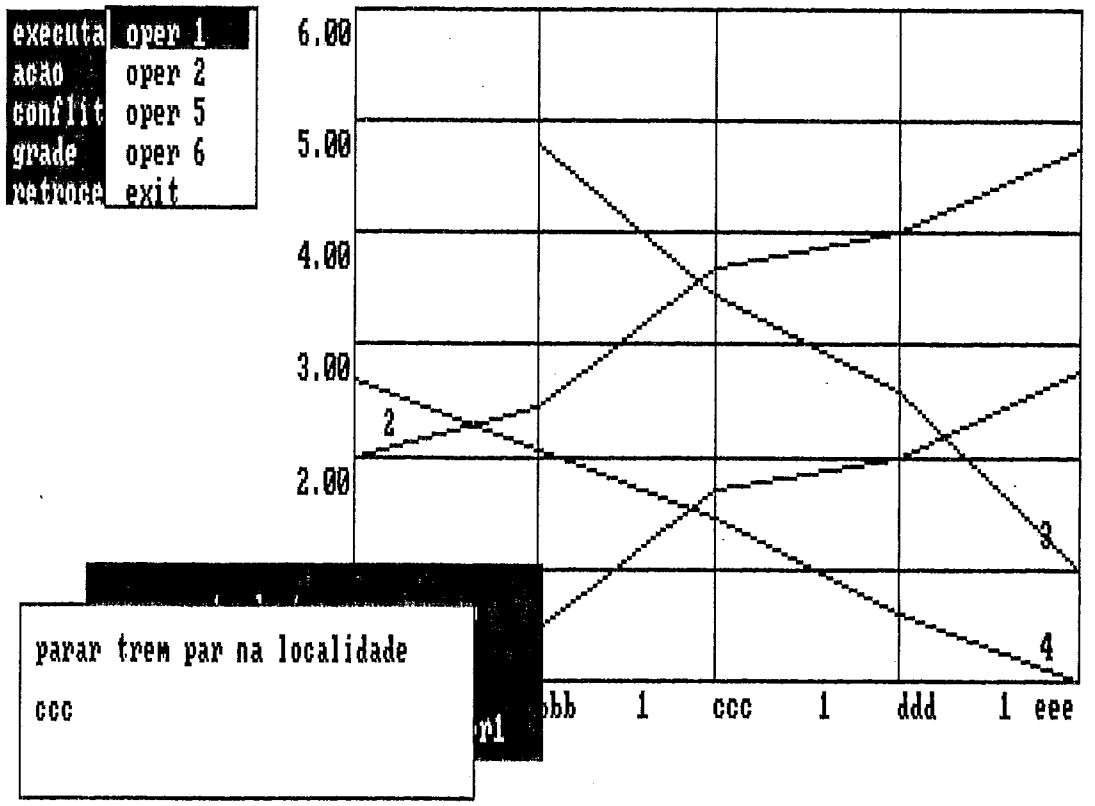


figura 15: geração de tempos de percurso aleatórios.

5 - O SIGT oferece a possibilidade de uso integrado dos sistemas de planejamento e simulação. Desta forma, é possível realizar a melhor simulação da ferrovia (atraves do planejamento ótimo) e utilizar o resultado para avaliações e comparações de performance de simulações voltadas para o treinamento;

6 - O modelo permite a geração de variáveis aleatórias obedecendo a várias distribuições de probabilidade (normal, exponencial, erlang, beta, gamma, lognormal, etc.) e classes de frequência, bem como de parâmetros "fuzzy" obedecendo a distribuições de possibilidade lineares (H. Tanaka e K. Asai [1984]) e do tipo L-R (D. Dubois e H. Prade [1980]), ver figura 15.

7 - O modelo de simulação, bem como o sistema de planejamento, permitem a realização de estatísticas a respeito da ocorrência de atrasos em patios de cruzamentos e seções da malha ferroviária, para cada trem em circulação. Entre outros parâmetros, o sistema fornece: atraso total acumulado, frequência absoluta, frequência relativa, atraso médio, variância, intervalos de confiança, valores máximos e mínimos

observados, etc. Possibilitamos, também , a realização de estatísticas baseadas em observações independentes, como exemplo: determinar a ocupação média de um pátio de cruzamento, através do método regenerativo (W.R. Franta [1977]).

Finalmente, queremos destacar que o modelo de simulação utiliza os mesmos componentes do SIGT utilizados pelas outras funções, ou seja: módulos de regras, mecanismos de inferência e explanação, memória de trabalho, estados, operadores, função de avaliação, heurísticas, "frames", menus, janelas, gráficos, grades, pilhas, tabelas, listas,etc.

- Monitoração :

De posse de um planejamento ou plano de ações sequenciais, que determinam as decisões operacionais em determinado setor da ferrovia, atendendo a um período de planejamento estipulado, podemos simular a monitoração deste planejamento identificando um intervalo de varredura ou um intervalo de monitoração que determinará a avaliação dos objetivos e o calculo dos atrasos acumulados, ver figura 16.

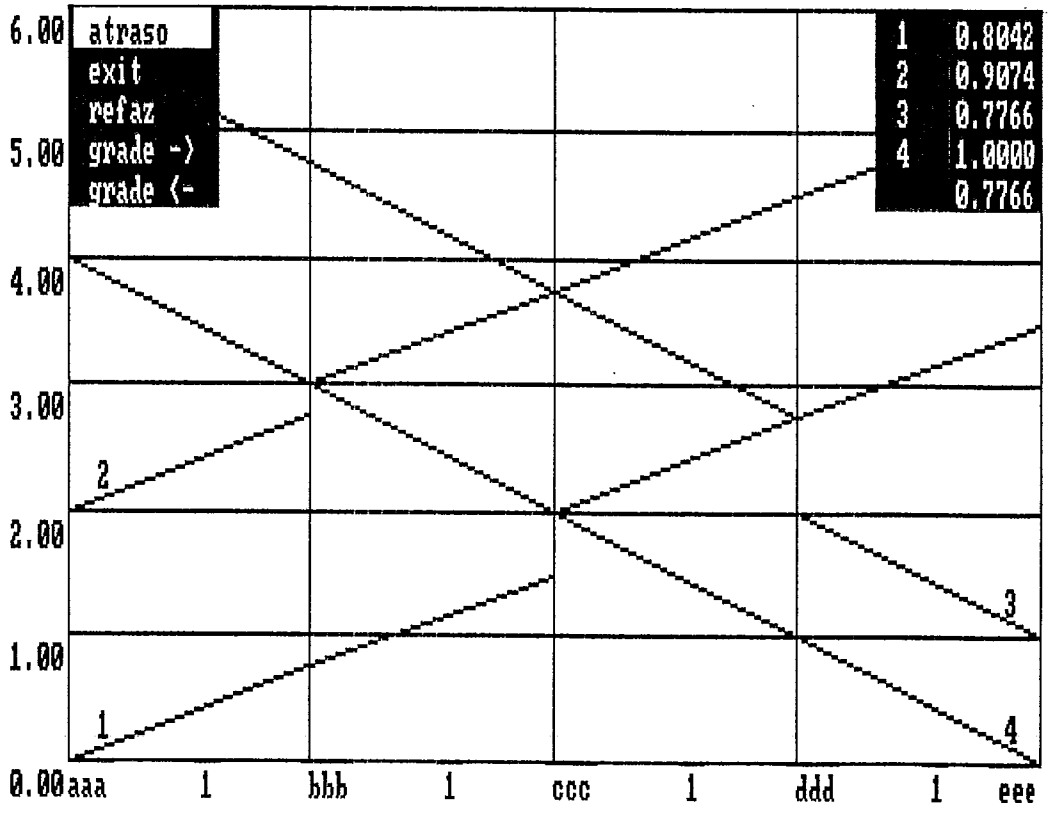


figura 16: plano de ações elaborado taticamente.

A monitoração avança, discretamente, a cada incremento do intervalo, ate deparar-se com a projeção de um conflito (o instante de ocorrência do conflito será no proximo intervalo), ver figura 18.

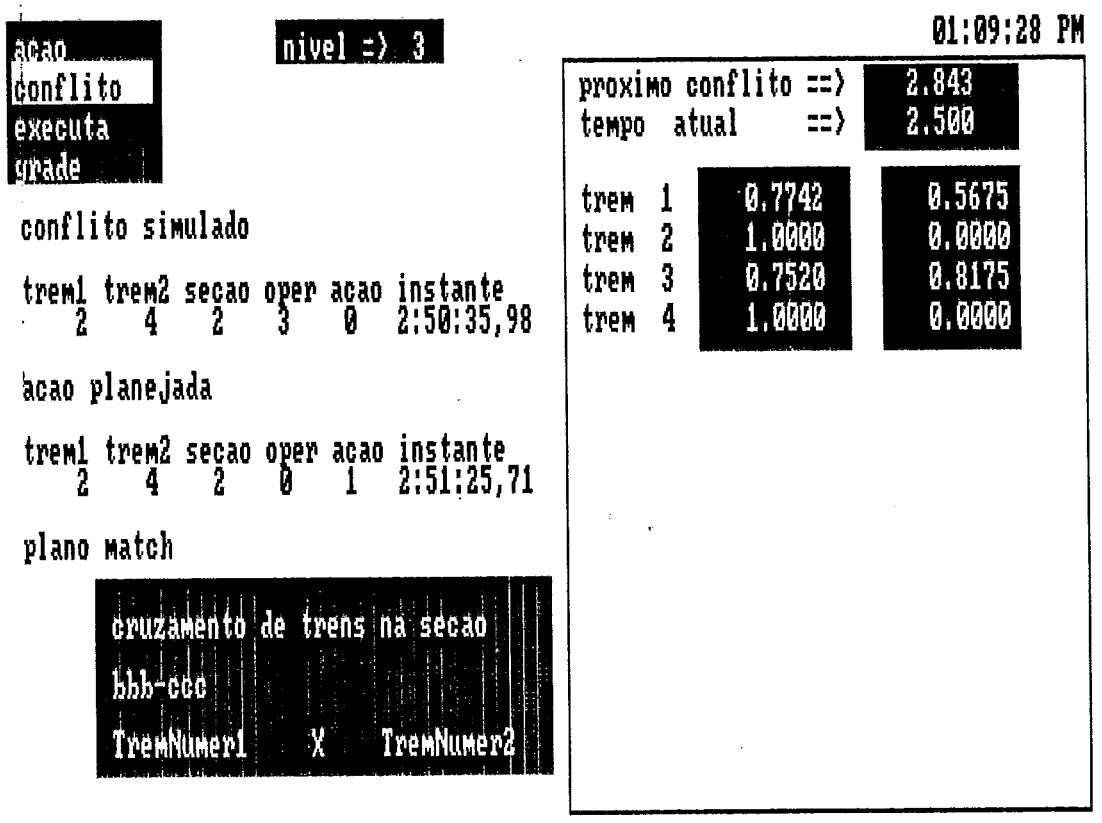


figura 17: situação de conflito detectada.

Neste instante, o modelo de monitoração verifica a viabilidade da ação planejada, confrontando-a com a situação real, ver figura 18. Caso a ação seja viável (coincide os trens e o ponto do conflito) a monitoração prossegue, caso contrário, é efetuado o replanejamento a partir do instante do último conflito solucionado, reavaliando as prioridades dinâmicas dos trens e reconstruindo as funções de pertinência e a tabela de tempos do estado planejado, ver figuras 19 e 20.

Para simularmos uma situação real de operação introduzimos no modelo uma variável que caracteriza o grau de imprecisão do sistema ,permitindo a geração de parametros "fuzzy" segundo a função de H. Tanaka e K. Asai [1984]).

A grande vantagem do modelo de monitoração e possibilitar os testes e consequente validação do gerenciamento e controle de trens pelo SIGT, sem a necessidade de realizarmos experimentos reais.

Também, durante a monitoração, é possível, com o uso de janelas e menus, a visualização progressiva de resultados gráficos (grades de trens) e numéricos (atrasos acumulados e valor dos objetivos) bem como, a descrição textual de ações e conflitos. Este modelo, a exemplo do modelo de simulação, partilha dos mesmos componentes do SIGT.

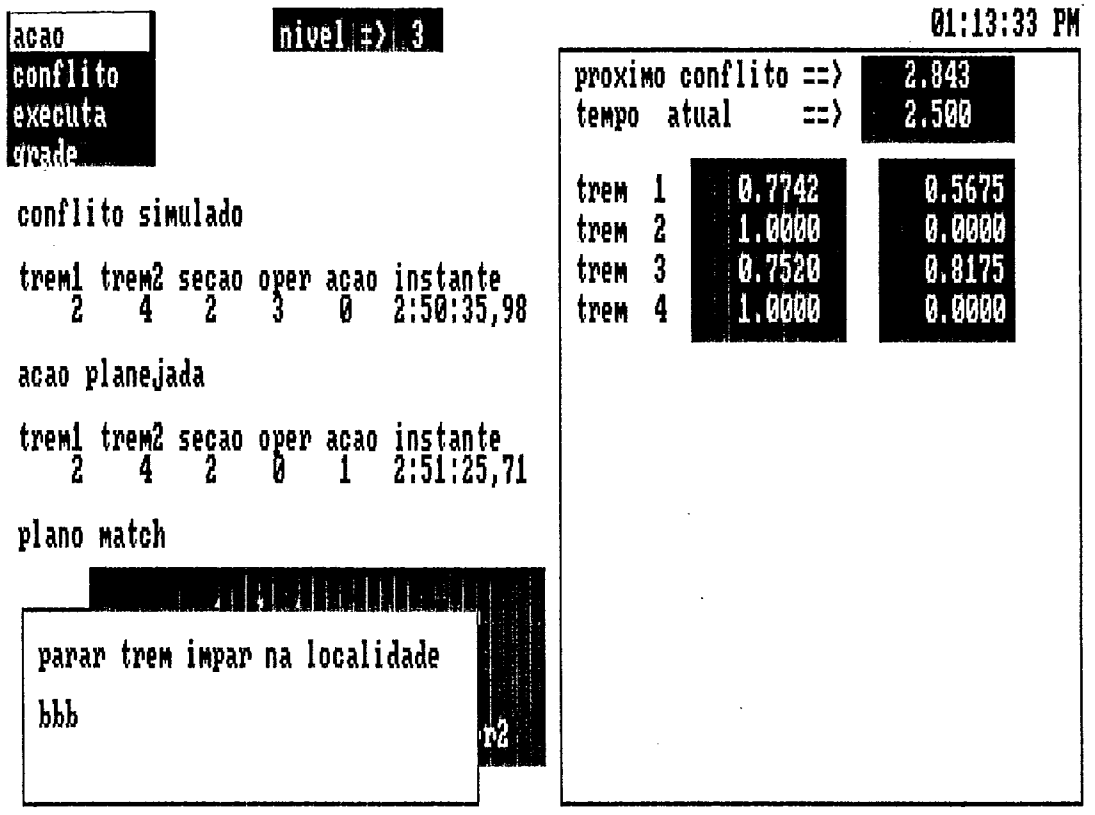


figura 18: ação programada (plano "match")

- Interface Amigável:

A interface amigável do sistema, permite ao usuário interagir com o mesmo, acessando e visualizando suas principais funções e componentes . Basicamente, esta interface compreende:

1 - Janelas para apresentação de textos e resultados numéricos;

2 - Menus (janelas com opções de escolha);

3 - Gráficos;

4 - Perguntas e respostas na forma interativa, e

5 - Rotinas para a manipulação de arquivos.

Alguns exemplos de utilização da interface amigável com o usuário incluem:

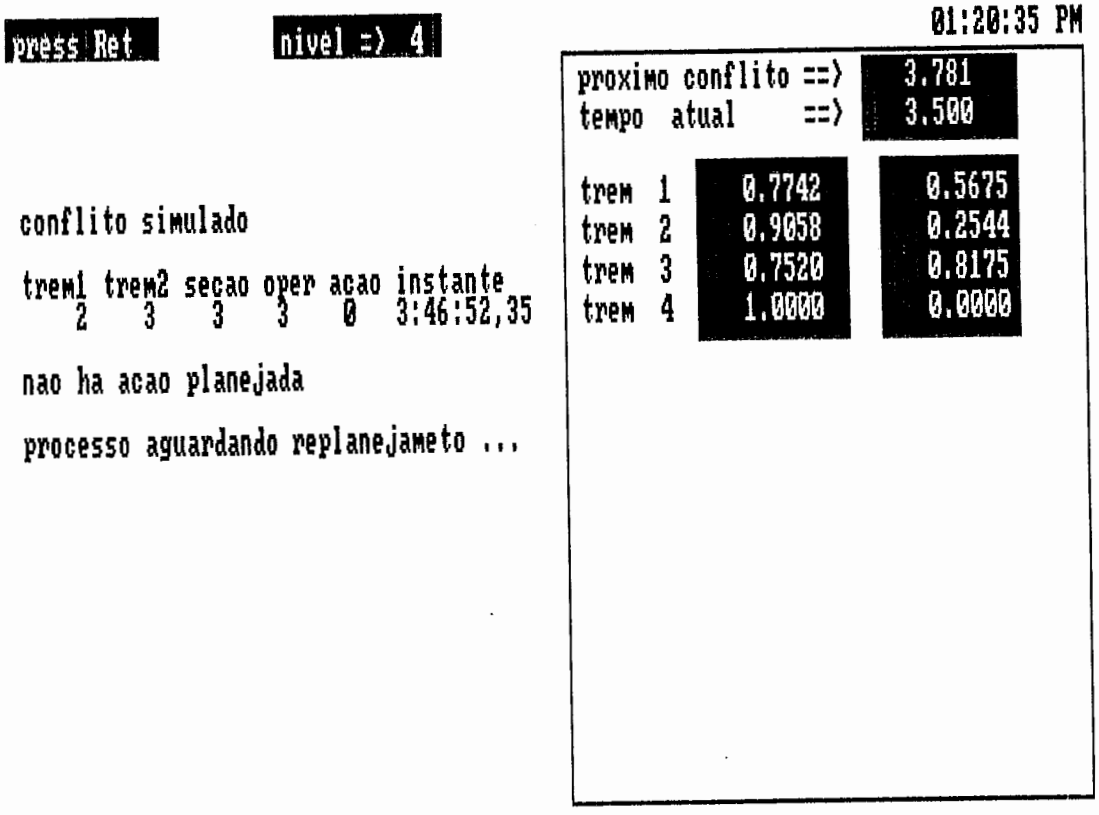


figura 19: replanejamento (plano falha).

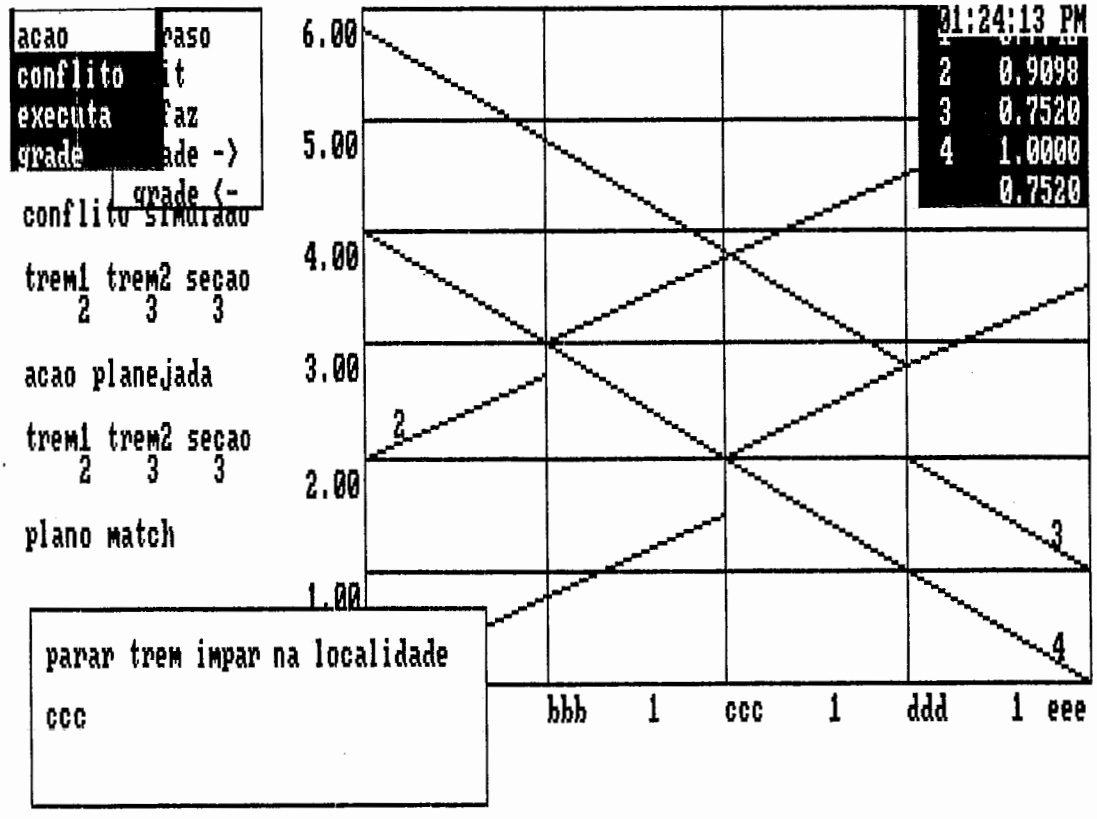


figura 20: ação reprogramada e visão parcial da grade de trens.

capacidade e funcionalidade incorporada ao SIGT. Entre os mais importantes e pioneiros sistemas , podemos destacar o sistema ISIS , " Intelligent Scheduling and Information System " desenvolvido no Instituto de Robótica da Universidade de Carneggie Mellon localizada em Pittsburgh, EUA (M.S. Fox [1982]), o qual será analisado em nosso trabalho.

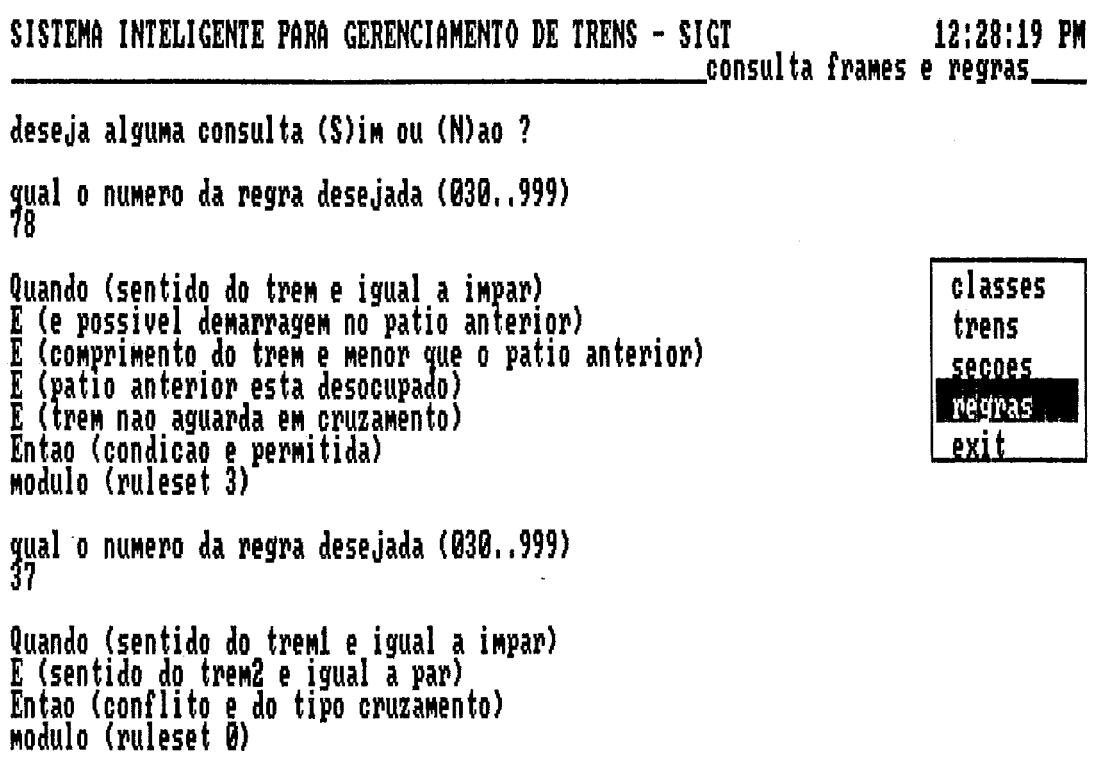


figura 22: visualização de regras.

2.2.5 - Propriedades

Entre as propriedades mais marcantes que um sistema inteligente deve apresentar destacam-se:

1 - Acessibilidade : os modelos que compõe o sistema devem ser acessíveis ao usuário;

2 - Extensibilidade : os modelos devem ser facilmente alterados pelo usuário;

3 - Consistência : os modelos devem prevenir a ocorrência de inconsistências quando forem expandidos, e, finalmente,

4 - Robustez : os modelos devem suportar variações e situações imprevisíveis.

. janelas para descrição de conflitos e ações na forma textual nos modelos de simulação e monitoração;

. janelas para apresentação de resultados numéricos na monitoração;

. menus para explanação de raciocínio no sistema de produção;

. menus para visualização e modificação de "frames";

. menus de controle da simulação;

. menus de controle da monitoração;

. menus de escolha de operadores;

. grafico das funções de pertinencia, ver figura 21;

. grade de trens com janela de opções;

. visualização da tabela de tempos;

. visualização de regras, ver figura 22;

. controle das funções principais do SIGT;

. opções de formulação e planejamento e

. arquivos com dados de trens, seções, classes, etc.

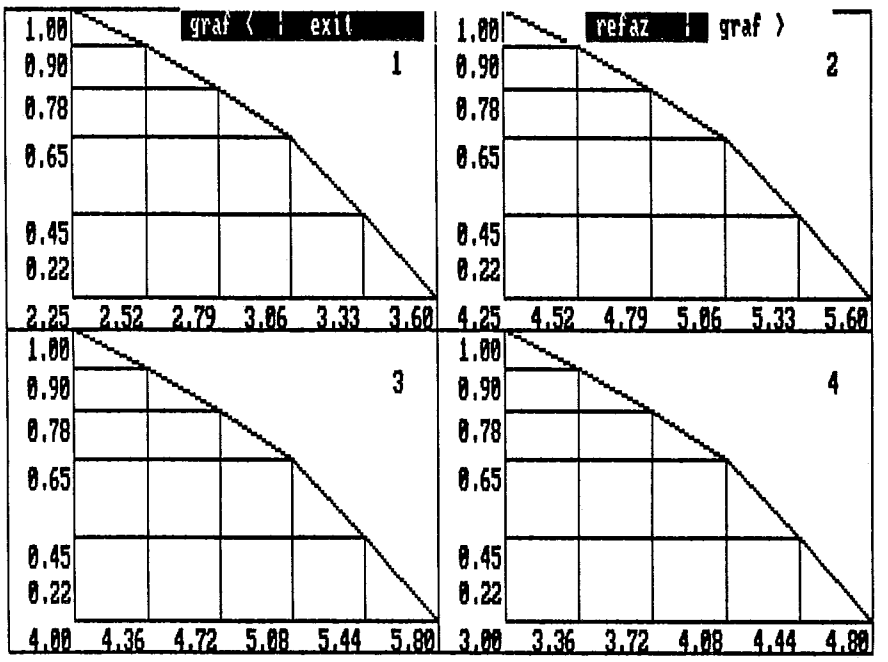


figura 21: funções de pertinência com atraso máximo estipulado em 60% do tempo de percurso teórico.

Recentemente, a maioria dos sistemas inteligentes voltados para as funções de planejamento e programação em tempo real possuem a

- Acessibilidade:

Quanto a propriedade de acessibilidade, o sistema SIGT proporciona a escolha e acesso ao modelo adequado . Por exemplo, o usuário, quando utiliza o SIGT, possui as seguintes opções quanto a modelagem do problema e a formulação de objetivos, ver quadros 2 e 3.

quadro 2: flexibilidade quanto a modelagem do problema:

função	formulação	algoritmo	raciocinio
planejamento	max-min	A*	usa sistema produção
		IDA*	usa sistema produção
	inequações	Busca Progressiva	usa sistema produção
simulação	discreta orientada pela solução conflitos	exame de atividades	usa sistema produção

quadro 3: flexibilidade quanto a formulação de objetivos:

agregação	calc. atraso	operador	reavaliação
agrega ou não objetivos por classes	internamente, percentual ou valor fixo	min	compensatória

- Extensibilidade:

Quanto a propriedade de extensibilidade, podemos destacar a facilidade que o SIGT apresenta para a manipulação das "frames". Como estas estruturas são implementadas segundos objetos, podemos criar, destruir e modificar, facilmente, qualquer seção, trem ou classe de trens da ferrovia. Esta propriedade, é de grande importância para sistemas que operam em tempo real, pois o programa deve suportar alterações na estrutura de dados em tempo de execução . Portanto, como

exemplo, ao longo do processo de planejamento podemos inserir um novo trem, fornecendo apenas suas informações próprias (que especializam o mesmo), já que as informações comuns são herdadas da classe a qual o mesmo pertence.

Também, devemos destacar a extensibilidade do conhecimento armazenado na forma de regras de produção, que pode ser ampliado com extrema facilidade, através da incorporação de novas heurísticas.

- Consistência e Robustez:

Quanto a propriedade de consistência, que afeta principalmente o conhecimento do sistema, o SIGT não apresenta problemas . Isto se deve, principalmente, ao fato, do sistema de produção ser organizado na forma hierárquica, reduzindo a sua complexidade (um módulo possui no máximo 30 regras).

Finalmente, queremos ressaltar a propriedade de robustez, caracterizada , principalmente, pela capacidade que um sistema inteligente em tempo real deve possuir, em termos de garantir respostas em determinado intervalo fixo de tempo, compatível com a necessidade da aplicação e com as imprevisões acarretadas pelo nível de imprecisão do sistema aplicativo. Esta propriedade, também e satisfeita pelo nosso sistema de controle e gerenciamento de tráfego de trens.

Como sugestão de leitura , indicamos o trabalho apresentado por (T.J. Laffey e outros [1988]) , onde é apresentado a descrição básica de inúmeros sistemas baseados em conhecimento, que operam em tempo real, incluindo aplicações nos setores aeroespacial, comunicações, controle de processos e robótica.

2.3 - Planejamento e Controle

2.3.1 - Hipóteses Fundamentais

O sistema SIGT possui como aliçerce básico, três hipóteses fundamentais que tornam viável o gerenciamento e controle inteligente de trens:

1 - É possível no sistema ferroviário realizar a operação de trens utilizando planejamento;

2 - É possível modelar o problema de despacho de trens como um sistema dinâmico baseado na realização de eventos discretos, permitindo, a partir da extensão de um formulação de "scheduling", obter a representação espaço-estado;

3 - É possível, a partir da representação espaço-estado, desenvolver um sistema inteligente que permita a realização do planejamento.

Quando nos referimos a possibilidade do planejamento, estamos admitindo a possibilidade de orientarmos a operação de trens com a utilização de planos de horários. A viabilidade desta hipótese, dependerá, principalmente, da confiabilidade e precisão do sistema ferroviário considerando-se, principalmente, o sistema de controle, o sistema de sinalização e os parâmetros que representam o tempo de percurso dos trens.

Seria ingenuidade de nossa parte, admitirmos a completa precisão do sistema ferroviário. Neste caso, seria possível a elaboração de planos ótimos envolvendo um ciclo completo de transporte. Por outro lado, a existência de alta imprecisão inviabiliza a definição de políticas de transporte,impossibilitando a otimização da circulação de trens.

A primeira hipótese, comprovada experimentalmente, levanta a possibilidade da realização de um planejamento dinâmico, que dependendo da imprecisão do sistema , pode assumir qualquer posição entre a realização do planejamento ótimo e a tomada de simples decisões localizadas (controle operacional).

Quanto a segunda hipótese, admitindo que o nome: sistemas dinâmicos baseados em eventos discretos, designam (G. Cohen, D. Dubois, J.P. Quadrat e M. Viot [1985]) uma classe de sistemas cuja conduta é completamente caracterizada pelo conhecimento dos eventos e atividades, podemos considerar a operação e circulação de trens como tal, já que foi possível a determinação de um modelo de " job-shop

scheduling" que incorpora o início e término dos eventos que caracterizam as atividades relacionadas a operação de trens. Este modelo permitiu a construção de uma representação espaço-estado, apoiada na utilização de tabelas de tempos, que representam o início de operação das atividades.

Quanto a terceira e última hipótese, foi possível estruturarmos um sistema físico simbólico através da representação espaço-estado do sistema dinâmico, o que possibilitou, com o uso de conhecimento intensivo, o desenvolvimento de um sistema inteligente voltado para o gerenciamento e controle de trens .

2.3.2 - Operação Primitiva do Despacho de Trens

- Grade de Trens:

Até pouco tempo, o despacho de trens foi caracterizado pelo uso do sistema de licenciamento do tipo "staff" elétrico ou telegráfico.

A operação destes sistemas (abertura e fechamento de chaves) é do tipo manual e localizada, não possibilitando, portanto, a centralização das decisões de controle. Tratando-se de sistemas mais primitivos, sua utilização e condicionada a ferrovias com baixa densidade de trens (intervalo de tempo maior entre eventos consecutivos), que possuem maior precisão devido a existência de tempos de percurso muito elevados e , também, de uma política operacional caracterizada por partidas de trens em horários pre-fixados.

Por causa destas particularidades , os sistemas primitivos utilizam com muita constância planos de horários ou grade de trens, para orientarem a solução de conflitos, considerando, neste caso, a total impossibilidade de serem tomadas decisões em tempo real.

Estes planos são elaborados a um nível tático de planejamento por métodos analíticos tradicionais ou modelados como um problema de "job-shop scheduling " segundo R. Szpigel [1972] . Entretanto, tratando-se de um problema de grande complexidade , os planos elaborados ficam seriamente afetados quanto a sua dimensionalidade. Szpigel, em seu trabalho,reporta a solução de problemas contendo no máximo 10 trens

envolvendo 5 seções de uma ferrovia, levando para a obtenção da solução ótima, aproximadamente 25 minutos em um computador de médio porte.

- Funções do Despachador:

Nos sistemas primitivos de despacho de trens a principal função do despachador é a elaboração da grade de trens determinando a solução de conflitos , já que as funções de controle ficam resumidas as operações físicas de abertura e fechamento de chaves. Ressalta-se ,também, a tarefa de comunicação das decisões (partida de um trem , condições de circulação, etc) as estações adjacentes.

2.3.3 - Operação com Controle de Tráfego Centralizado (CTC)

- Descrição do CTC:

O sistema CTC tem como função comandar e controlar o movimento de trens na malha ferroviária. A introdução deste sistema permitiu maior eficiência na operação de licenciamento de trens, pois o movimento dos mesmos passou a ser controlado por um sistema de sinalização automático, situado em um centro localizado.

O movimento de trens é monitorado remotamente pelo despachador, através de painéis sinóticos situados no centro localizado. A mudança das posições de chave é feita automaticamente pelo despachador, através do manuseio de chaves, botões ou teclado alfa-numérico.

Além de permitir um controle e gerenciamento centralizado, o CTC contribui para a segurança do movimento através de um sistema de intertravamento que verifica a seguranca operacional das decisões tomadas pelo despachador.

Atualmente, o sistema CTC atingiu um grau de automatização bastante elevado, permitindo a identificação de trens através de códigos e visualização do posicionamento no painel sinótico.

As funções lógicas em tempo real do sistema (aquisição de dados, acionamento do painel e interface com o console de comando) estão sendo executadas por controladores programáveis (R.D. P. Barbosa e outros [1986]), ligados ao sistema de transmissão de dados,

por micro-ondas, trazendo grande flexibilidade e eficiência ao sistema CTC.

Existem, também, na configuração do sistema, impressoras cuja função principal e registrar todas as decisões efetivadas pelo despachador, bem como os eventos que determinam a circulação de trens (entrada no pátio, saída do pátio, etc).

- Funções de Controle:

Entre as principais funções do despachador, no sistema CTC, podemos destacar:

. monitorar o movimento de trens pela observação do painel sinótico ou terminal de video;

. preparar o plano de rotas;

. reconhecer e resolver situações de conflito;

. executar os comandos de controle sequencial (mudança de chaves);

. registrar os eventos operacionais;

. cumprir o planejamento (estratégico e tático) estabelecido,e

. comunicar as decisões operacionais ao controle de tráfego adjacente.

Considerando a possibilidade de automação do registro de eventos operacionais, as atividades mais importantes que restam ao despachador estão associadas as tarefas de planejamento e tomada de decisões.

Sendo assim, achamos totalmente factível a automação parcial do despacho de trens pelo sistema SIGT.

2.3.4 - Por que Planejar ?

- Otimização Requer Projeção:

Conforme observações feitas anteriormente, a possibilidade da realização de um planejamento dinâmico que oriente a tomada de decisões no controle operacional de trens dependerá dos níveis de precisão e confiabilidade do sistema ferroviário. Desta forma , quanto mais precisos forem os parâmetros do sistema e quanto mais confiável for o sistema CTC e o sistema de sinalização, melhor será a execução do

planejamento.

Ou seja, se houver alta imprecisão no sistema, a execução de um planejamento viável fica seriamente limitada, havendo necessidade quase imediata de replanejamento, afetando, desta forma , a otimalidade temporal dos planos de horários.

Quando citamos a otimalidade temporal dos planos, nos referimos a dimensão do problema ao longo do tempo, ou a profundidade do planejamento na árvore de busca do problema. Ou seja, em um nível imediato tratamos da solução de um conflito local, entretanto, a medida em que aprofundamos a árvore de busca a elaboração do plano cresce na dimensão do tempo.

Portanto, podemos garantir que a otimização de planos de ações requer a projeção temporal dos conflitos. Desta forma, é possível ao SIGT selecionar o melhor plano de horários que atenda as metas e estratégias estabelecidas, analisando o efeito acumulado de todas as decisões ou ações intermediárias.

Walter Reitman [1982], discute a aplicação de técnicas de Inteligência Artificial nos sistemas de suporte a decisões, descrevendo um programa de computador elaborado para jogar GO. O jogo GO, bastante comum aos orientais, envolve uma quantidade de análise e planejamento tático e estratégico muito superior ao jogo de xadrez. D.J.H. Brown e S. Dowsey [1979] estimam que o número de possíveis jogadas do GO e da ordem de 10^{700} comparado a estimativa de 10^{120} posições alternativas do jogo de xadrez.

O GO é jogado entre duas pessoas em um tabuleiro quadriculado (19x19), com um conjunto de 42 peças numeradas (21 brancas e 21 pretas) que determinam a ordem do movimento. O jogo consiste de jogadas alternativas (movimento de peças no tabuleiro) com o objetivo fundamental de capturar as peças do adversário. Uma peça é capturada se for cercada por peças inimigas.

O jogo possui uma quantidade imensa de variações e estratégias podendo, de acordo com a disposição das peças, serem criadas estruturas como cadeias, grupos, pontes ,territórios, etc. O programa desenvolvido para jogar GO possui três componentes principais:

1 - Um componente que analiza e descreve a situação do jogo, atualizando a representação das estruturas denominado DESCRIBE;

2 - Um componente que constroi e avalia as alternativas de movimento denominado CONSTRUCT, e

3 - Um componente que desenvolve linhas de raciocínio estimando as consequências táticas e estratégicas, denominado PROJECT.

O programa funciona, basicamente, efetuando o ciclo DESCRIBE-CONSTRUCT, até que seja necessário a definição de uma linha de raciocínio. Neste ponto, o componente PROJECT e invocado.

Uma alternativa de movimento definida pelo componente CONSTRUCT deve levar em conta a solução de problemas urgentes como ataque ou defesa imediata, ou jogadas táticas e estratégicas, como reforço de posições, contra-ataque, etc.

Como já foi observado, o sistema DESCRIBE-CONSTRUCT desenvolve uma representação coerente com a hierarquia e interrelação dos elementos do jogo e determina a solução de problemas imediatos ou propõe jogadas cujos méritos estão condicionados a futuras possibilidades. Portanto, somente com o uso do sub-sistema PROJECT é possível observar a frente, e, consequentemente, avaliar tática e estratégicamente o efeito destas jogadas, orientando a evolução do jogo segundo linhas de raciocínio.

A estrutura deste programa, elaborada para jogar GO, orientou bastante o desenvolvimento do SIGT. Podemos distinguir no nosso sistema os três componentes e respectivas funções, quando efetuamos o gerenciamento e controle do tráfego ferroviário sob a orientação de planos.

Neste sentido, podemos constatar que o modelo de monitoração avalia a cada ciclo (intervalo de monitoração) os valores das funções de pertinência e atualiza as informações relativas a base de conhecimento (podemos considerar este processo como a jogada do adversário, cuja adversidade e proporcionada pela imprecisão do sistema). Quando defronta-se com um conflito (problema urgente) o sitema SIGT avalia e interpreta o tipo de decisão proposto pelo plano e propõe uma alternativa de jogada ao despachador. Caso no processo de avaliação , não exista alternativa viável de jogada (nível insatisfatório de algum objetivo ou inviabilidade da ação planejada) o sistema efetua em tempo hábil o replanejamento (planejamento tático e estratégico)

determinando, novamente, uma linha de raciocínio.

A diferença básica entre o jogo GO e o controle e gerenciamento do tráfego de trens é relativo as funções exercidadas pelo componente CONSTRUCT. No jogo GO, existem múltiplas situações de conflito que são resolvidos segundo prioridades baseadas na urgência ou importância dos mesmos. No controle e gerenciamento de trens, os conflitos são atacados na ordem cronológica, pois trata-se de um sistema em tempo real cuja ordem de prioridade é ditada pelo tempo.

- Linhas de Raciocínio:

Tradicionalmente, o controlador de CTC não trabalha sob a orientação de um planejamento, principalmente, devido a impossibilidade de elaboração de planos em tempo real. Desta forma, podemos dizer que o controlador trabalha sob o efeito do imediatismo das ações, não desenvolvendo, portanto, linhas de raciocínio. Em entrevistas com controladores de tráfego, observamos que os controladores que conseguem melhor desempenho de circulação (aumento da capacidade de trafego) trabalham programando pontos de cruzamento. Ou seja, estes controladores projetam a solução de pelo menos um conflito.

No sistema SIGT, segundo testes realizados, temos a capacidade de gerar planos de horários ótimos com a solução de 40 conflitos em aproximadamente 45 segundos em computadores PC com processador INTEL 80286, ou em 35 segundos com processador INTEL 80386SX, ambos sem o coprocessador numérico.

Isto significa obtermos a melhor solução dentro de um universo que abrange, aproximadamente, 10^{12} alternativas de planos, ver figura 23.

Segundo a literatura ferroviária e informações de especialistas, o fator humano (interferência do despachador) na solução de conflitos provoca uma variação em torno de 25% da capacidade de tráfego de uma ferrovia, principalmente, quando o volume de tráfego ultrapassa 25 pares de trens diários.

Desta forma, achamos que o sistema proporcionará, em média, um aumento na capacidade de tráfego superior ou em torno de 25% , considerando a utilização do conhecimento de um despachador habilidoso associado a capacidade de desenvolvimento de linhas de raciocínio.

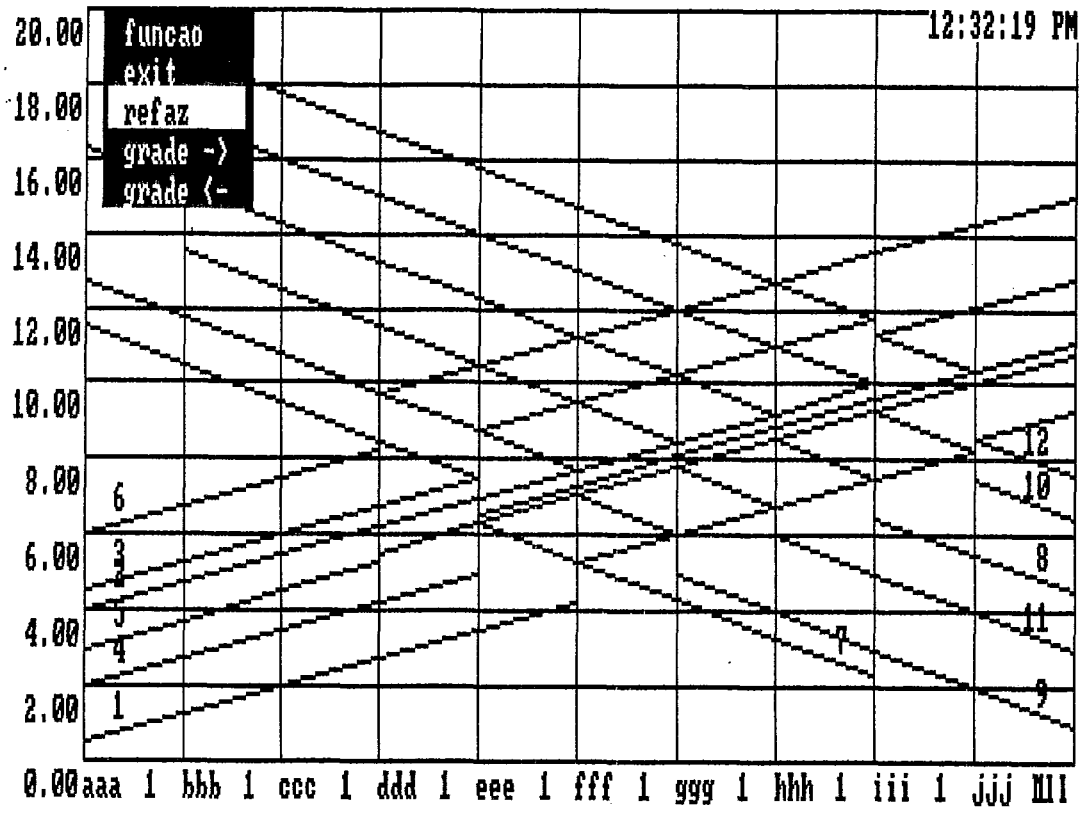


figura 23: plano ótimo com solução de 40 conflitos.

- Utilização de Padrões:

Uma importante consideração a ser feita é que alguns controladores utilizam uma forma de reconhecimento de padrões, ou de situações similares, na solução de alguns conflitos. Algumas situações bem características e facilmente detectáveis, como por exemplo: medidas para prevenção de bloqueios, prioridade para onda de trens nos conflitos, etc., foram incorporadas ao SIGT na forma de produções (regras).

A utilização de reconhecimento de padrões ou situações similares voltada para a solução de problemas ou tomada de decisões é muito discutida na literatura, sendo uma das formas mais comuns de raciocínio humano. Como exemplo, podemos citar observações feitas por Hebert Simon [1981], a respeito da realização de partidas simultaneas entre um mestre sênior de xadrez e vários concorrentes.

Neste ambiente de jogo, onde o tempo de resposta e limitado a poucos segundos, torna-se impossível para o jogador o desenvolvimento de

linhas de raciocínio, mesmo por que, seria necessário o desenvolvimento de uma linha de raciocínio diferente para cada oponente. Neste caso, o mestre sênior joga quase intuitivamente, tomando decisões provenientes do reconhecimento de situações similares ou padrões. É provado que o desempenho do mestre diminui bastante, entretanto, o mesmo é capaz de manter o nível de bom jogador , vencendo várias partidas.

Um exemplo marcante da utilização de reconhecimento de padrões em problemas de comando e controle e mostrado no trabalho de (Se-Youg Oh [1986]) que descreve um modelo de controle "on line" para sistemas de potência voltados para a análise da segurança. Tradicionalmente, a solução deste problema e obtida analiticamente atraves da solução de um modelo de fluxos de potência para determinada lista de contigências (possíveis falhas no sistema). Os resultados , ou fluxos de carga apurados, são analisados e a segurança do sistema verificada comparando-se os fluxos de potência em cada "link" da rede modelada com valores admissíveis.

Como a solução de um único problema requer uma quantidade de tempo imcompatível com a operação em tempo real do sistema , Se-Young optou pelo desenvolvimento de um conjunto de padrões provenientes de estados operacionais pré-analisados . Desta forma, o modelo reconhece a segurança de um estado particular do sistema em tempo real através do reconhecimento de padrões ou estados operacionais similares .

No controle de tráfego ferroviário, seria impossível a utilização de reconhecimento de padrões como forma de planejamento. Isto porque, o problema de reconhecimento de padrões se tornaria mais complexo que o próprio problema de planejamento. Apenas como curiosidade, vamos admitir a realização do planejamento para um setor da ferrovia contendo 24 seções e aproximadamente 16 trens. Considerando a existência de dois blocos em cada seção , teríamos, portanto, a possibilidade de existência de 48!/32! alternativas de padrões diferentes , admitindo uma aproximação ou agrupamento de padrões coerente com o rastreamento discreto do CTC, mas totalmente imcompatível com a precisão necessária ao planejamento.

Portanto, achamos que a utilização de reconhecimento de situações operacionais similares, só se justifica nas decisões de

conflito localizadas , podendo, também, orientar na elaboração de planos pequenos com poucas alternativas de configuração (D. Wilkins [1980]).

No SIGT, a utilização de padrões, na forma de regras, serve, principalmente, para reduzir o fator de ramificação da árvore de busca, possibilitando a elaboração de planos complexos em tempo real.

- Conhecimento Associado:

N.M. Morris e W.B. Rouse [1985] discutem em seu trabalho o efeito do tipo de conhecimento incorporado ao controlador na tarefa de tomada de decisões ou soluções de problemas relativos a sistemas dinamicos (controle de trafego, controle de processos, etc.).

Segundo Morris e Rouse , com o consequente avanço tecnológico dos sistemas de controle, passou-se a exigir dos operadores uma capacidade mental (raciocínio rápido, reconhecimento de padrões, solução de tarefas, elaboração de planos) muito acima da habilidade físico-motora. Os autores discutem no trabalho a utilização de três tipos básicos de conhecimento na tarefa de controle de uma planta de tanques interconectados (operação de valvulas, operação de bombas, monitoração de alarmes, controle de níveis, etc.), a saber:

1 - Conhecimento mínimo a respeito das funções, objetivos e operação do sistema;

2 - Conhecimento procedural indicando o que fazer em determinadas situações operacionais (caderno de procedimentos), e

3 - Conhecimento a respeito dos princípios e equações que governam o problema .

Considerando, que a importância de cada forma de conhecimento ,depende do tipo de sistema a ser controlado, vamos analizar a utilização dos mesmos no que tange ao controle e gerenciamento do tráfego ferroviário.

Ao que parece, a forma mais provável de conhecimento utilizada pelos despachadores é o conhecimento procedural, ou seja, o despachador percebe situações operacionais similares e aplica determinados procedimentos . O conhecimento a respeito dos princípios e equações que governam o problema de sequenciamento e programação, não parece trazer melhoria ao controle e gerenciamento . Entretanto, o conhecimento

mínimo a respeito, principalmente, dos objetivos e restrições operacionais, é importante na perseguição de metas e garantia da viabilidade das decisões.

Porém, se todas as formas de conhecimento não forem utilizadas para a elaboração de linhas de raciocínio, o processo decisório cai no imediatismo do controle operacional (tomada de decisões localizadas). Desta forma, mesmo que o despachador possua as três formas de conhecimento e tome as melhores decisões localizadas, progressivamente, a otimalidade dos planos (sequenciamento de decisões) ficará seriamente prejudicada.

Podemos imaginar, o que seria de um jogador de xadrez que para tomar as suas decisões, observa-se apenas o passado do jogo, não projetando suas decisões raciocinando para frente. Mesmo que este jogador possuísse um conhecimento comparável a de um mestre sênior, a sua atuação seria sempre corretiva, imediatista e certamente medíocre.

A respeito do papel importante dos planos no processo de solução de problemas , segundo David Wilkins, vamos repetir uma frase do grande mestre Alexander Kotov:

" É melhor seguirmos um plano consistente, mesmo que este não seja o melhor deles, do que não utilizarmos nenhum."

- Dimensionalidade do Planejamento:

A dimensão do plano ao longo do tempo afeta a otimalidade temporal e obriga a integração de planos consecutivos. Neste sentido é possível manter a mesma linha de raciocínio conservando as metas e critérios, reavaliando prioridades e reconstruindo os objetivos. Entretanto, qual é o período ideal de planejamento ?. A resposta óbvia seria aquele período de tempo no qual não fosse necessário a realização do replanejamento. Este período, naturalmente, depende do comportamento do sistema.

Um período ideal de planejamento seria aquele equivalente ao turno de trabalho do despachador, ou seja, em torno de 6 horas. Entretanto, se considerarmos uma ferrovia com alta densidade de tráfego com, aproximadamente, 25 pares de trens diários ou um intervalo médio de

30 minutos entre trens, e uma seção de controle com, aproximadamente, 25 pátios e tempo médio de percurso entre patios em torno de 20 minutos, teríamos a possibilidade de gerarmos 10^{24} planos alternativos (correspondentes a solução de aproximadamente 80 conflitos).

Entretanto, este planejamento seria intratável em tempo real. Nestas mesmas condições, o SIGT seria capaz de propor um planejamento de 4 horas correspondentes a solução de aproximadamente 40 conflitos em um tempo de computação razoável, conforme foi visto anteriormente.

No exemplo acima, mencionamos outro componente da dimensionalidade dos planos que é aquele que define o trecho ou seções da ferrovia sob planejamento. Obviamente, este trecho deve comportar o controle de tráfego setorizado, que possui tamanho variável. No caso da Linha do Centro (RFFSA/SR-3 - Juiz de Fora), pode conter de 20 à 30 seções.

A dimensão linear do plano obriga a integração de planos adjacentes, envolvendo os setores da malha ferroviária, determinando , desta forma, a necessidade de coordenação das interfaces.

É interessante observarmos que o esforço computacional e a utilização de memória no SIGT é uma função polinomial (linear) do numero de seções sob planejamento se mantivermos o numero de conflitos, através da redução proporcional do período de planejamento e vice-versa.

2.4 - Aplicabilidade do Sistema SIGT

2.4.1 - Concepção Alternativa de Planejamento Ferroviário

Nesta seção, vamos procurar descrever a capacidade e potencialidade do sistema, no que tange a tomada de decisões relativas ao planejamento ferroviário, excetuando as funções de gerenciamento e controle, que já foram analisadas previamente.

Antes de citarmos as aplicações proporcionadas pelo sistema , vamos apresentar os níveis principais do planejamento ferroviário , segundo o contexto de A.A. Assad [1980].

Segundo Arjang Assad, o planejamento estratégico na ferrovia

delineia decisões que envolvem grande investimento de capital e aquisição de recursos, produzindo efeitos num horizonte de longo prazo. Por exemplo:

- Modificação de "lay-out" da malha ferroviária;
- Definição de fluxos de transportes (metas de produção) e
- Localização de pátios de classificação e cruzamento.

A um nível tático de planejamento, cujas decisões provocam um efeito a médio prazo, Arjang cita:

- Composição, itinerário e roteamento de trens;
- Definição de políticas operacionais (pátios e linha) e
- Definição de tamanho de trens.

Finalmente, a um nível operacional de planejamento, onde as decisões refletem o aspecto dinâmico da operação ferroviária , com efeito quase imediato, temos:

- Definição de rotas e tabelas de horários;
- Políticas de prioridade;
- Políticas de despacho e
- Distribuição de vagões vazios e locomotivas.

2.4.2 - Ferramenta para Realização do Planejamento

Seguindo o contexto proposto por Assad, vamos identificar a capacidade do SIGT relativa ao estudo, análise e avaliação das decisões estratégicas, táticas e operacionais que compreendem o planejamento ferroviário.

O modelo de simulação e planejamento desenvolvido pelo SIGT, segundo Assad, está voltado para o estudo e análise de decisões que afetam a capacidade do sistema ferroviário , mais especificamente, a capacidade de tráfego.

Dentre os fatores que determinam a capacidade de uma ferrovia, podemos destacar três grupos principais, segundo a visão de Assad:

- Fatôres relacionados ao "lay-out" da malha ferroviária e a decisões estratégicas:

. número de linhas;

. capacidade de pátios de cruzamento;

. estado da via permanente;

. distância entre pátios de cruzamento;

. bloqueio automático (tempos de "headway");

. tipo de sinalização e

. geometria e perfil da linha.

- Fatôres relacionados a característica dos trens e a decisões táticas:

. pêso e potência dos trens;

. comprimento dos trens;

. velocidade dos trens (tempos de percuso) e

. densidade de trens.

- Fatôres relacionados a operação ferroviária e a decisões operacionais:

. política de prioridades;

. política de despachos;

. circulação de trens de serviço;

. intervalos de manutenção;

. conflitos (ultrapassagens, cruzamentos, etc);

. parada de trens e

. balanceamento do tráfego .

Todos estes fatores são considerados no sistema de planejamento e simulação desenvolvido para o SIGT, possibilitando a avaliação de decisões estratégicas, táticas e operacionais na ferrovia quanto a interferência na capacidade de tráfego.

No nosso sistema, abordamos o problema da capacidade ferroviária segundo a ótica mais racional , ou seja, proporcionamos o estudo interrelacionado dos fatores e decisões, de forma paramétrica . Os parâmetros adotados pelo modelo estão associados a eficiência do sistema , avaliando o nível de serviço proporcionado (satisfação dos objetivos) e os tempos de atraso acumulados.

2.4.3 - Ferramenta para Operações Simuladas:

Finalmente, queremos destacar a potencialidade do SIGT para a realização de operações simuladas. Com a utilização simultânea dos modelos de planejamento e simulação, é possível orientarmos o sistema para o desempenho de funções voltadas para o treinamento de controladores.

A flexibilidade proporcionada pelo uso combinado das duas funções básicas, permite a avaliação momentanea de decisões, explanação de raciocínio, simulação de produções (regras), visualização e modificação das estruturas de informação e acompanhamento de planos pré-elaborados.

É possível, também, através da monitoração simulada, predizer o efeito da operação sob planejamento quanto a ocorrência de imprecisão nos parâmetros do sistema. Através do uso de simulações podemos criar alternativas de operação e avaliar a eficiência e consistência de novas regras de julgamento (heurísticas), ampliando, desta forma, o conhecimento próprio ("expert") do sistema.

Capítulo III
Geração de Planos de Horários

3.1 - Considerações Iniciais

O problema de despacho de trens consiste, a um nivel tático de planejamento, em determinar o sequenciamento ótimo de diversos trens em uma malha ferroviária. A solução ótima do problema pode ser traduzida em um plano de horários ou " schedule", devendo, portanto, ser cumprida dentro do período que abrange o planejamento realizado. A formulação matemática mais comum, segundo (B. Szpigel [1972]) consiste em equacionarmos o problema de sequenciamento de trens como um problema de "job-shop scheduling" e utilizarmos para a sua solução um técnica de programação linear relaxada, seguido de um método "branch and bound".

Nesta formulação, a função objetiva do problema retrata a otimização global do sistema (minimiza os tempos de atraso) atendendo diferentes níveis de prioridade para diferentes tipos de trens. Estas prioridades são estáticas e determinadas pelo "staff" da ferrovia.

A um nível de planejamento operacional, podemos considerar o problema de despacho de trens como um problema de tomada de decisões envolvendo a solução de conflitos, caracterizando-se como um problema dinâmico, mal estruturado, e que exige a tomada de decisões em tempo real.

Podemos caracterizar o problema de despacho de trens como um problema mal estruturado e impreciso, observando vários de seus aspectos. Por exemplo, em relação ao seu objetivo, sabemos que, na realidade, existem vários objetivos e critérios que possam influenciar a escolha de uma boa solução. Neste caso, podemos associar a cada trem ou grupo de trens um nível de satisfação relativo ao atendimento realizado. Desta forma, ao utilizarmos uma modelagem "fuzzy" (C.V. Negoita[1981]) para o problema, daremos um tratamento similar aos objetivos e restrições do mesmo .

Ao que parece, uma formulação em forma de inequações que permita a identificação de uma solução viável para o problema, atendendo as restrições e aos níveis de satisfação estabelecidos, é mais eficaz e natural que o processo de otimização convencional, citado inicialmente.

Outro aspecto que reforça a estrutura mal definida do problema de despacho de trens é a imprecisão relativa aos parâmetros que representam o tempo de percurso dos trens. Para tentarmos resolver este problema podemos introduzir o conceito de toria de possibilidades (L.A. Zadeh [1978]) associando uma função de distribuição de possibilidade que restringe, de uma certa forma, os valores assumidos pelos parâmetros do problema.

Alem das restrições normais ao problema de sequenciamento , temos no problema de despacho de trens restrições que incorporam estratégias superiores, restrições físicas específicas e restrições características de um sistema em tempo real, que levam ao surgimento de novos cenários e novas situações.

Para tratarmos estas novas restrições, parece mais razoavel a utilização de um conjunto de regras na forma de um sistema de produção, utilizando um método de raciocínio que seja orientado de forma progressiva , segundo o encadeamento de regras, de modo a restringir o espaço de soluções do problema.

De uma forma geral, podemos admitir que o controlador trabalha segundo planos de rota ou de horários. Os controladores procuram, na medida do possível, seguir o raciocínio destes planos que são elaborados a um nível tático de operação. Caso alguma restrição ou nível de satisfação do problema seja fortemente violado, como por exemplo: atraso de um trem, o controlador aciona a elaboração de um novo plano com novas prioridades. Portanto, podemos considerar que as prioridades são dinâmicas e ajustaveis as situações reais do controle.

A elaboração de um plano de horários pode ser conseguida através da resolução de um problema de programação "fuzzy", envolvendo determinado setor da malha ferroviária, utilizando um processo de busca em um grafo de estados segundo uma função heuristica.

Em geral partimos de uma solução global (envolvendo vários conflitos) inviavel e, através de um esquema "branch and bound", resolvemos os conflitos de forma sucessiva até alcançarmos uma solução viável, não necessariamente otima, estabelecendo um limite superior para o problema. O desenvolvimento "branch and bound" é equivalente ao

processo de construção de uma arvore de busca , estabelecendo uma hierarquia de conflitos relativa ao tempo de ocorrência dos eventos. E importante ressaltarmos, que a função de avaliação a ser utilizada, deve refletir a característica dinâmica e imprecisa do problema.

É fundamental no processo acima a geração de planos de horários viáveis, pois o tempo utilizado para a pesquisa, dependerá do tempo de resposta do sistema. Portanto, este procedimento deverá ser otimizado da melhor forma possível. É importante, também, que este procedimento forneca uma solução viável, a qualque tempo, mesmo que esta solução não seja de boa qualidade. Neste particular, podemos considerar a hierarquia de conflitos na árvore de busca segundo a ordem natural de ocorrência , pois, neste caso, haverá sempre a garantia de produzirmos planos de horários viáveis.

Como cada plano de horário envolve apenas um setor da malha ferroviária , teremos, também, que considerar a coordenação e comunicação de diferentes sub-sistemas e de diferentes controladores na execução do despacho de trens.

3.2 - Descrição do Problema

3.2.1 - Introdução

Conforme visto anteriormente, podemos considerar o problema de despacho ou sequenciamento de trens como um problema de planejamento de horários, ou problema de "scheduling", com restrições de ordenamento deferidas por relações de precedência.

Para caracterizarmos melhor o problema de sequenciamento de trens, vamos introduzir algumas considerações que se farão necessárias ao desenvolvimento do modelo matemático.

Seja um trecho ferroviário (figura 24) qualquer, constituído por uma sucessão de segmentos definidos como seções.

Podemos associar as seções deste trecho, a um conjunto de máquinas em uma linha de processamento sequencial, referente a um processo decisório de multi-estágios. Cada seção ou máquina é caracterizada por uma certa capacidade e um conjunto de tempos de

processamento equivalentes aos tempos de percurso dos trens.

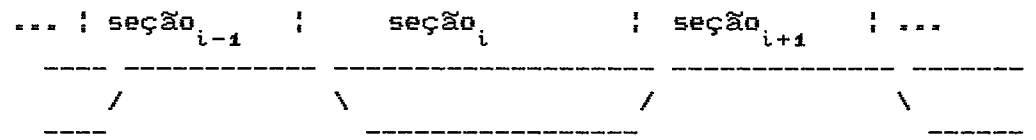


figura 24: trecho ferroviário (s_{i-1}, s_{i} , s_{i+1}).

O problema clássico de sequenciamento, ou problema de "scheduling" do tipo "job-shop", pode ser definido como (R.L. Graham, E.L. Lawler, J.K. Lenstra e A.H. Kan [1977]):

" Dados "p" jobs e "m" máquinas e os tempos de processamento de cada job em cada máquina, encontre uma ordem na qual os "p" jobs são processados nas "m" maquinas atendendo a um certo objetivo."

Para definirmos melhor o problema de sequenciamento de trens, devemos considerar alguns aspectos relativos ao conjunto de restrições e a função objetiva do mesmo, como veremos a seguir.

3.2.2 - Restrições Estruturais

- Restrições no Processamento de Jobs:

No problema de "job-shop scheduling" temos as seguintes características relevantes:

. nenhum processamento de job pode ser interrompido;

. cada job só pode ser processado por uma máquina de cada vez;

. a ordem de processamento dos jobs nas máquinas não é necessariamente comum.

Portanto, considerando que o processamento de um job consiste de um conjunto de atividades sequenciais, existe um ordenamento lógico para o processamento do mesmo, que deve ser respeitado.

Este ordenamento pode ser considerado como o itinerário de um trem no trecho ferroviário em questão, e estabelece um conjunto de

restrições que serão conhecidas como restrições de sequenciamento.

Além das restrições de sequenciamento, teremos restrições de precedência e restrições relativas a capacidade de filas nos postos de cruzamento ou ultrapassagem (pátios de espera). As restrições de precedência , são restrições que estabelecem uma ordem de processamento dos jobs, para cada máquina . Entretanto, dada a natureza dinâmica do problema, veremos que o procesamento de jobs se faré segundo a ordem de chegada , salvo o caso de ocorrência de conflitos ocasionados pelas restrições relativas a capacidade das seções.

3.2.3 - Tipos de Conflitos

Basicamente, temos três tipos de conflitos:

- Conflitos de Cruzamento:

. linha singela:

Neste caso, trens que tenham sentido contrário disputam uma seção com capacidade inferior ao número dos mesmos. O exemplo mais comum é quando dois trens disputam uma seção singela da ferrovia (figura 25).

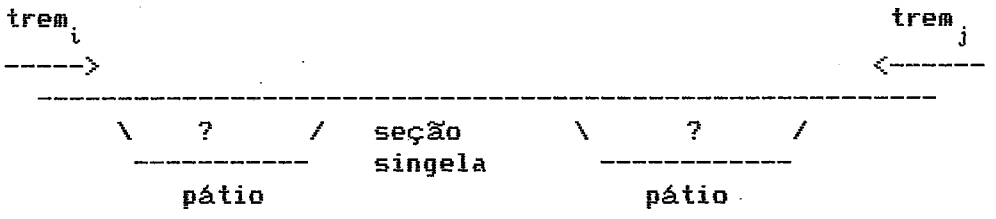


figura 25: conflito de cruzamento linha singela.

Conforme podemos verificar na figura 25, este tipo de conflito gera, a primeira vista, duas possibilidades de solução, ou seja, podemos desviar (parar) o trem "i" ou o trem "j", nas seções adjacentes, de linha dupla, que são, neste caso, consideradas postos (pátios) de cruzamento ou ultrapassagem.

. linha dupla:

Neste caso, cada trem deve tomar uma linha , sendo que o trem mais prioritário tem preferência pela linha principal . A parada de algum trem sera necessária quando a sua linha estiver ocupada de modo a não permitir o avanço. No exemplo da figura 26, verificamos que o trem "j" deve aguardar a passagem do trem "i", para em seguida efetuar o cruzamento com o trem "k". A transferência de uma linha para outra e efetuada pelo uso dos travessões.

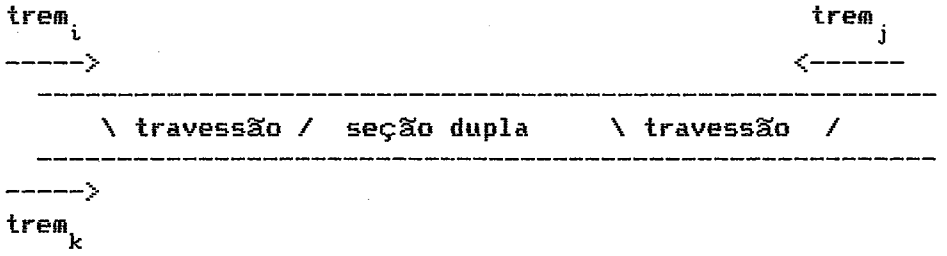


figura 26: conflito de cruzamento linha dupla.

- Conflitos de Ultrapassagem:

. linha singela:

Neste caso, figura 27, trens que tenham mesmo sentido disputam uma seção com capacidade inferior ao número dos mesmos. Portanto, para se realizar a ultrapassagem é necessário que o trem mais lento e de menor prioridade aguarde o processamento do trem mais rápido e de maior prioridade, ocorrendo, desta forma, uma modificação das relações de precedência no problema.

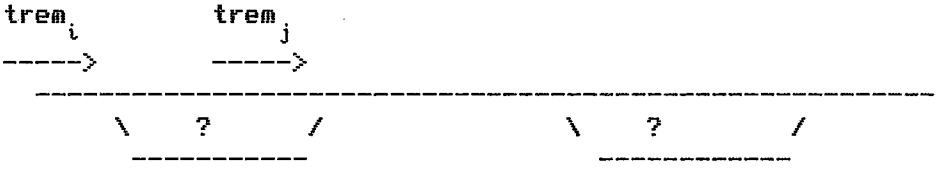


figura 27: conflito de ultrapassagem linha singela.

Este tipo de conflito, também gera duas possibilidades de

solução, ou seja, podemos desviar o trem "j" no pátio anterior ou no pátio posterior da seção onde ocorreria o conflito.

. linha dupla:

Neste caso, a ultrapassagem é realizada normalmente, com o trem prioritário tendo preferência pela linha principal. Na hipótese de haver algum trem em sentido contrário (figura 28) a ultrapassagem e retardada ate a seção posterior.

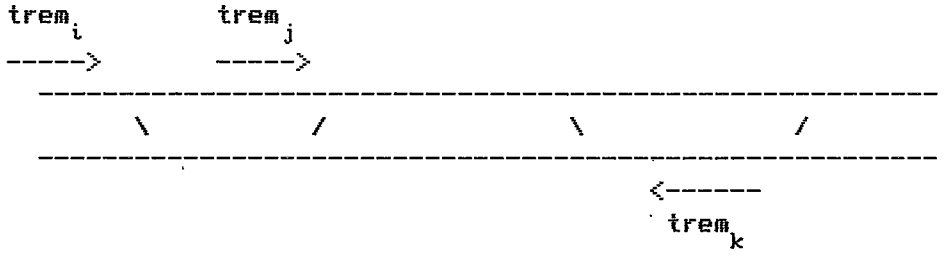


figura 28: conflito de ultrapassagem em linha dupla.

- Manutenção do "Headway":

Considerando, também, que uma máquina ou seção da ferrovia pode admitir o processamento ou tráfego de vários jobs ou trens , segundo a sua capacidade, teremos um conflito ocasionado pela manutenção do "headway" , ou pela manutenção de um intervalo de tempo entre dois trens consecutivos de mesmo sentido , definido para cada seção da ferrovia, segundo a situação descrita na figura 29.

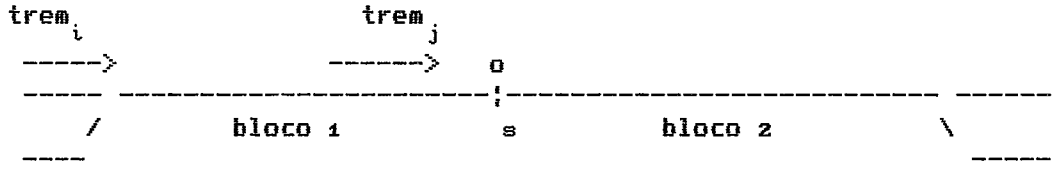


figura 29: manutenção de "headway".

Neste caso, o trem "i" deve aguardar o trem "j" passar pelo sinal de bloqueio "s" de modo a liberar o bloco anterior .

É importante ressaltar que na hipótese de um trem ter que aguardar a liberação da seção seguinte, dada as restrições físicas do problema, o mesmo deve aguardar na seção anterior, ampliando, desta forma, o tempo de processamento na seção.

Considerando que a existência de conflitos ocorrem, principalmente, nas seções singelas, vamos considerar, inicialmente, a não ocorrencia de conflitos em postos de cruzamento ou ultrapassagem. A viabilidade das restrições de sequenciamento ficarão garantidas se admitirmos tempos adicionais ou tempos de "set up" nos tempos de processamento dos jobs.

Convém, também, esclarecermos que a liberação de uma seção pode ocorrer de duas formas:

. nos conflitos de cruzamento:

Neste caso, é necessário o processamento total do job, garantindo o percurso do trem até o posto onde se efetuará o cruzamento.

. nos conflitos de ultrapassagem ou manutenção de "headway":

Neste caso, a mesma seção pode suportar o tráfego de dois ou mais trens, necessitando apenas que se guarde um tempo pré-determinado entre dois trens consecutivos (os trens possuem o mesmo sentido). Este tempo pré-determinado, conhecido como "headway", garante a segurança do movimento, já que o mesmo é superior ao tempo máximo de frenagem dos trens que circulam na ferrovia. O tempo de "headway" é característico do sistema de sinalização existente na ferrovia sendo operacionalizado pela implantação dos sinais automáticos .

Vimos até agora, que o problema de sequenciamento de trens se enquadra na categoria de problemas do tipo "job-shop scheduling" , com restrições de sequenciamento e precedência, considerando a ocorrência de conflitos.

Podemos, também, classificar o problema em questão como um problema de "scheduling" dinâmico, no sentido de que a chegada de jobs ou trens é semelhante a um processo de filas .

Trata-se, ainda, de um problema de "scheduling" estocástico, pois existe uma certa aleatoriedade na chegada dos jobs, bem como no tempo de processamento dos mesmos.

Entretanto, como veremos adiante, podemos, a um nível operacional de planejamento, tratar o problema em questão na forma determinística, considerando a existência de uma imprecisão ou nebulosidade nos parâmetros que estabelecem os tempos e o início de processamento dos jobs.

Segundo (S.C. Graves [1981]) o problema de "schedulling" se torna, na prática, um problema de "reschedulling", pois uma dificuldade maior que estabelecermos um "schedulling" apropriado e determinarmos as constantes revisões requeridas em um sistema dinâmico com incertezas, de forma a garantirmos a continuidade da otimalidade do mesmo e a viabilidade do plano de horários.

- Restrições na Disponibilidade das Máquinas:

No problema clássico de "scheduling" todas as máquinas devem estar a disposição durante o período de planejamento. Desta forma, qualquer ocorrência de evento que provoque a paralização da atividade de processamento não é permitida.

Entretanto, no sistema ferroviário , a ocorrência de eventos que afetam ou até mesmo impedem a circulação de trens é muito comum. A ocorrência destes eventos é totalmente aleatória, não sendo portanto, possível, a um nível tático de planejamento, considerarmos a existência dos mesmos.

Outro fato importante, esta relacionado com a manutenção prévia ou reparo das máquinas. Neste caso, é possível sabermos com antecedência o período relativo a manutenção das mesmas. No problema de sequenciameto de trens podemos considerar os períodos de manutenção a priori, ou, posteriormente, através da utilização de janelas nos "schedules" gerados.

3.2.4 - Medidas Clássicas da Função Objetiva

Seja :

$\bar{C}$ - o tempo médio de término da última operação;

$\bar{F}$ - o tempo médio total de processamento;

$\bar{W}$ - o tempo médio total de atraso ou espera e

$\bar{L}$ - o tempo médio de esgotamento do tempo total de processamento previsto,

referentes ao processamento total de todos os jobs no problema de "scheduling".

Existe, segundo (R. Bellman, A.O. Esogbue e J. Nabeshina [1982]), neste tipo de problema, uma classe de funções objetivas que conduzem para a mesma solução ótima. Portanto, podemos enunciar o seguinte teorema:

No problema de sequenciamento, se uma solução é ótima para alguma das funções objetivo: $\bar{C}$, $\bar{F}$, $\bar{W}$ OU $\bar{L}$, então, também é ótima para as outras.

Prova:

Podemos definir o tempo médio de término da última operação, como:

$$\bar{C} = \sum_{i=1}^{n} r_i/n + \sum_{i=1}^{n} p_i/n + \sum_{i=1}^{n} W_i/n = \bar{r} + \bar{p} + \bar{W} , \qquad (1)$$

onde:

$$\bar{p}_i = \sum_{j \in R(i)} p_{i,j} , \qquad (2)$$

se refere ao tempo médio de processamento de cada job em cada máquina "j", $j \in R(i)$, sendo $R(i)$ o conjunto de máquinas relativos ao job de índice "i";

$$\bar{W}_i = \sum_{j \in R(i)} W_{i,j} , \qquad (3)$$

se refere ao tempo média de espera de cada job em cada máquina j, $j \in R(i)$;

r_i é referente ao tempo de início do processamento de cada "job" i, i=1,...n , e

"n" se refere ao número total de "jobs".

Podemos, também, definir o tempo médio total de processamento , como:

$$\bar{F} = \bar{p} + \bar{W} ; \qquad (4)$$

Como os valores de $\bar{r}$ e $\bar{p}$ são constantes , é equivalente minimizarmos $\bar{F}$, $\bar{C}$ ou $\bar{W}$.

O tempo médio de esgotamento do tempo total de processamento previsto é igual a:

$$\bar{L} = \bar{C} - \bar{d} , \qquad (5)$$

onde:

$$\bar{d} = \frac{\sum_{i=1}^{n} d_i}{} , \qquad (6)$$

e d_i se refere ao tempo total de processamento previsto para o job "i", i=1,...,n.

Das expresssões [1], [4] e [5], derivamos:

$$\bar{F} + \bar{r} = \bar{L} + \bar{d} , \qquad (7)$$

como $\bar{d}$ e uma constante, e equivalente minimizarmos $\bar{F}$, $\bar{C}$, $\bar{W}$ ou $\bar{L}$ ■

Quando formulamos a função objetivo do problema de scheduling" com expressões relativas a $\bar{F}$, $\bar{C}$, $\bar{W}$ ou $\bar{L}$, podemos atribuir pesos de acordo com a prioridade dos jobs.

No problema de sequenciamento de trens a atribuição de pesos na

função objetivo seria relativa a importância dos trens ou , mais especificamente, a classes de trens, cujo objetivo principal é o transporte de mercadorias de mesmo gênero.

Da teoria de "scheduling" (R.Bellman, A.O. Esogbue e J. Nabeshina [1982]) sabemos que , para um intervalo fixo de tempo, o numero médio de jobs no sistema e proporcional ao tempo médio total de processamento ou ao tempo médio de término da última operação.

Desta forma, para o nosso problema , podemos estabelecer a seguinte relação:

$$\bar{F} = \bar{N} / n \ , \qquad (8)$$

onde:

$\bar{N}$ se refere ao número médio de trens no sistema e

n se refere ao número médio de trens que iniciam a operação por unidade de tempo.

Portanto, segundo B. Szpigel [1972], se minimizarmos a função objetivo clássica sem pesos, estaremos minimizando o numero total de trens no sistema e, consequentemente, o tempo total de atraso.

3.2.5 - Grade de Trens

Ao invés de representarmos o "schedule" no gráfico "Gantt Charts", que estabelece a ordem na qual "n" jobs são processados em "p" máquinas, optamos pela utilização de um diagrama espaço-tempo que nos fornece uma melhor visualização da solução de conflitos bem como do sequenciamento das operações.

Neste diagrama, conhecido como grade de trens, representamos no eixo horizontal a linha férrea, destacando os pontos de mudança caracterizados pelos pátios de cruzamento ou desvios e travessões, e no eixo vertical a escala de tempo relativa ao período de planejamento a ser realizado. Os trens ficam representados por funções que mapeam pontos que indicam o instante de tempo associado a respectiva localização. Desta forma, é possível identificar a posição de cada trem

, a cada instante, ao longo da linha férrea sob planejamento.

É possível, também, visualizarmos as decisões relativas a solução dos conflitos já ocorridos, bem como a projeção dos próximos conflitos potencialmente existentes.

3.3 - Formulação Matemática

3.3.1 - Notações e Formulação Clássica

Como sabemos, cada trem deve percorrer uma rota , caracterizada por um conjunto de seções sucessivas, ou seja:

$$R_i = (s_j) \quad \text{com} \quad j \in R(i). \tag{9}$$

Chamando de $t_{i,j}$ o início de processamento do job "i" na máquina "j", ou o instante no qual o trem ocupa a seção , e $p_{i,j}$ o tempo de processamento do job "i" na máquina "j" , ou o respectivo tempo de percurso na seção, podemos formular as restrições de sequenciamento do problema na forma:

$$t_{i,j} \geq r_i \quad \text{p/ } \forall \text{ i e j relativo a primeira seção ,} \tag{10}$$

$$t_{i,j} + p_{i,j} \leq t_{i,j+1} \quad \text{p/ } \forall \text{ i e j relativo as demais seções e} \tag{11}$$

$$t_{i,j} \geq 0 \text{ , p/ } \forall \text{ i e j , i=1,...n e j} \in R(i). \tag{12}$$

Vamos admitir que haja a ocorrência de um conflito entre dois trens "i" e "k" na seção singela "j" da ferrovia. Neste caso para garantirmos a não ocorrência do conflito é necessário introduzirmos as seguintes restrições:

$$t_{i,j} + x_{ik,j} \cdot p_{i,j} \leq t_{k,j} + x_{ki,j} \cdot M \text{ ,} \tag{13}$$

$$t_{k,j} + x_{ki,j} \cdot p_{k,j} \leq t_{i,j} + x_{ik,j} \cdot M \text{ ,} \tag{14}$$

$$x_{ik,j} + x_{ki,j} = 1 \text{ e} \tag{15}$$

$$x_{ik,j} \text{ , } x_{ki,j} \in \{0,1\} \text{ .} \tag{16}$$

O conjunto de restrições acima se aplica a qualquer trecho e a qualquer pares de trens da ferrovia, sempre que houver a ocorrência de conflitos. A constante M deve ser suficientemente grande para garantir a viabilidade das restrições , assegurando as restrições de precedência e as restrições de capacidade das seções.

Como as variáveis de decisão só podem assumir valores do conjunto {0,1}, teremos sempre na resolução de um conflito:

$$t_{i,j} \leq t_{k,j} + M \quad e \qquad (17)$$
$$t_{k,j} + p_{k,j} \leq t_{i,j}$$

ou

$$t_{k,j} \leq t_{i,j} + M \quad e \qquad (18)$$
$$t_{i,j} + p_{i,j} \leq t_{k,j} \ ,$$

estabelecendo, desta forma, uma condição de disjunção nas restrições que determinam a precedência de trens.

No primeiro caso , podemos dizer que o trem "k" precede o trem "i", significando que o trem "i" deve aguardar a liberação da seção "j". No segundo caso, temos a situação contrária.

Caso na ocorrência do conflito os trens "i" e "k" tenham mesmo sentido, o valor dos tempos de processamento necessários a liberação da seção "j" são equivalentes aos tempos de manutenção de "headway".

Na prática, o tempo decorrente da liberação da seção "j" e ocupação da seção "j+1" não é igual a zero. Concordando com as considerações feitas na seção 3.2.3, podemos acrescentar um tempo de "set up" equivalente a permanência do trem nos postos de cruzamento ou ultrapassagem às restrições [11]. Mesmo nas situações em que as ultrapassagens ou cruzamentos se realizam em movimento, a liberação e ocupação das seções não se realizam simultaneamente.

Na hipótese de algum trem ter que permanecer por algum tempo em algum posto de ultrapassagem ou cruzamento (neste caso denominaremos posto de serviço) acrescentamos, da mesma forma, o tempo de permanência devido, às restrições [11].

A formulação clássica do problema de sequenciamento de trens ,

conforme apresentada acima, foi feita, inicialmente, por (B. Szpigel[1972]).

3.3.2 - Problema Mal Estruturado

A um nível de planejamento operacional, devemos fazer algumas observações importantes a respeito da formulação matemática apresentada para o problema de sequenciamento de trens a um nível tático de planejamento.

Observando, que o objetivo básico da solução deste problema e proporcionar a geração de plano ou tabelas de horários, é natural que estes planos se aproximem o máximo da realidade.

Como visto anteriormente, dada a natureza dinâmica e imprecisa do problema, não foi possível estabelecermos uma representação apropriada para alguns parâmetros necessários a elaboração de um plano eficiente de horários que orientará a tomada de decisão dos controladores durante o gerenciamento e controle dos trens.

- Imprecisão no Sistema :

O instante de início de operação dos trens bem como o tempo de percurso nas diversas seções não podem ser determinados com precisão. Na verdade sabemos que , ao longo de uma série histórica, a taxa de chegada dos trens obedece a uma distribuição aleatória de Poisson e os tempos de percurso a alguma distribuição de Erlang de grau elevado.

Entretanto, sabemos, intuitivamente, que a inexatidão dos parâmetros associada a ausência , deficiência ou má interpretação da informação, está intimamente relacionado a necessidade e importância dos mesmos, no sentido de que, quanto menor for o tempo existente entre a obtenção da informação que determina um parâmetro e a sua utilização, maior será a possibilidade de trabalharmos com a informação e , consequentemente, com uma base de conhecimento corretos.

Portanto, tratando do desenvolvimento de um sistema cujo objetivo e fornecer subsídios para um gerenciamento e controle mais eficiente em tempo real, é possível que tenhamos um grau de certeza

maior em relação aos parâmetros envolvidos na elaboração dos planos de horários, se trabalharmos, sempre, com as informações mais recentes.

Segundo A.N.S. Freeling[1980], a teoria da probabilidade e de grande utilização nos sistemas onde existe um grau de incerteza ,associado a aleatoriedade, elevado.

Entretanto, a aleatoriedade esta relacionada, geralmente, a ausência de informações, e, consequentemente, a falta de evidência em relação a ocorrência de hipóteses. Desta forma, no nosso problema , talvez seja mais correto tentarmos associar a inexatidão ou incerteza dos parâmetros a imprecisão, relativa a deficiência ou mau uso da informação, e fazermos uso da teoria de possibilidades (L.A. Zadeh [1978]).

Esta questão, relativa a existência de imprecisão ou incerteza aleatória em parâmetros do sistema, que afetam a análise ou tomada de decisões, foi discutida por A.N.S. Freeling[1980] e R. Jain[1976]. Segundo R.E. Bellman e L.A. Zadeh [1976] a ocorrência de aleatoriedade pode ser corretamente modelada utilizando-se a teoria de probabilidade, enquanto que , a ocorrência de imprecisão, só é modelada corretamente pela teoria de "fuzzy sets".

Como exemplo, suponha que tenhamos a informação de que um trem irá partir daqui a duas horas. De fato, existe uma grande possibilidade do trem partir, exatamente, daqui a 2 horas. Uma possibilidade um pouco menor do trem partir em torno de duas horas, e assim por diante. Entretanto, se a ocorrência de algum evento provocar um adiamento da saída do trem, digamos por 30 minutos, teremos uma nova informação que altera o horário da saída do mesmo.

Outro exemplo, se refere a duração dos tempos de percurso dos trens. Se os últimos 20 trens tiveram um tempo de percurso, em uma seção qualquer da ferrovia, em torno de um determinado valor, existe uma grande possibilidade do proximo trem ter um tempo de percurso parecido se os fatores e condições que afetam o percurso de trens nesta seção não forem modificados. Entretanto, se antes do percurso deste trem, tivermos a informação de que choveu e , dado que chova, o rendimento dos trens cai em 20%, teremos, sem dúvida, uma possibilidade muito grande do tempo de percurso ser em torno de 20% maior do que os tempos anteriores.

- Restrições de Capacidade nas Filas de Espera:

Na formulação apresentada, não consideramos a viabilidade das restrições de capacidade em todas as seções da ferrovia. Conforme já mencionado, consideramos a resolução de conflitos , somente, nas seções singelas. Entretanto, nas demais seções, denominadas postos de cruzamento ou ultrapassagem, não é considerada a possibilidade de ocorrência de conflitos.

Caso haja a ocorrência simultânea de conflitos nas seções singelas e nos postos imediatamente adjacentes, teremos a ocorrência de um conflito sem solução ou bloqueio na ferrovia.

De um modo geral, as seções que servem como postos de cruzamento, ultrapassagem ou serviço na ferrovia são dimensionadas de forma a garantir a não ocorrência de bloqueios. Entretanto, quando o tráfego de trens se torna um pouco elevado, e comum o esgotamento de capacidade de alguns postos, principalmente, quando se utiliza postos de cruzamento ou ultrapassagens como postos de serviço.

- Problema Multi-Objetivo e Multi-Critério:

Na análise da função objetiva do problema clássico de "scheduling", verificamos a possibilidade de minimizarmos o tempo total de atraso do sistema se não considerarmos a existência de pesos. Este objetivo é bem razoável, quando não damos prioridades a execução de jobs . No sistema ferroviario, este objetivo seria válido se tivéssemos um único tipo de carga na ferrovia (todos os trens seriam iguais), não havendo, portanto, necessidade de atribuirmos prioridades diferentes.

Entretanto, na maioria das vezes, temos no sistema ferroviário a existência de classes de trens diferentes, com objetivos e metas de produção específicos.

De fato, ao respondermos o questionário, verificamos que o objetivo maior é atendermos grupos de trens de mesma classe em faixas de horários admissíveis, atendendo a níveis de satisfação estabelecidos por circunstâncias reais de operação e gerenciamento.

Desta forma, dada a característica dinâmica e hierárquica do problema de controle e gerenciamento de trens, verificamos, também, que

as prioridades das classes de trem variam conforme o desempenho dos mesmos; podendo, também, serem alteradas, respeitando regras e estratégias superiores estabelecidas.

Portanto, conforme já mencionado na introdução deste capítulo, parece mais provável estabelecermos um caráter multi-objetivo para a geração de planos, definindo um conjunto de objetivos e desenvolvendo funções de pertinência que possam avaliar o nível de satisfação ou cumprimento dos mesmos.

- Conflitos de Ultrapassagem:

A formulação apresentada por Szpigel não considera a possibilidade de conflitos de ultrapassagem. Neste caso , teremos a ocorrência deste tipo de conflito , sempre que, em algum trecho "j" da ferrovia, dois trens "i" e "k", de mesmo sentido, satisfizerem as inequações :

$$t_{i,j} > t_{k,j} \qquad (19)$$

e

$$t_{i,j} + p_{i,j} < t_{k,j} + p_{k,j} + H_j \qquad (20)$$

onde, H_j se refere ao tempo de "headway" na seção "j".

Neste caso, projeta-se a ultrapassagem do trem "k" pelo trem "i" na seção "j".

3.3.3 - Procedimento Computacional

- Algoritmo "Branch and Bound":

A resolução do problema de sequenciamento de trens, envolvendo todas as possibilidades de conflito pode ser conseguida através de um algoritmo "branch and bound" (B. Szpigel[1972]) que é o algoritmo mais comumente utilizado para a solução de problemas do tipo "job shop schedulling", podendo citar E. Balas[1969], H. Greenberg[1968], J. Charlton e C. Death[1970], entre outros; variando em relação as regras de ramificação , aos critérios de poda e a determinação de limites,

segundo características específicas das aplicações.

A solução inicial, que fornece um limite inferior ao problema, e conseguida através da relaxação das restrições de integralidade e utilização da técnica de programação linear. Entretanto, esta solução, não sera viável, pois, devido a relaxação das restrições de integralidade, as variáveis do tipo 0-1 apresentarão valores que vao inviabilizar, na solução de um conflito, a condição de disjunção das restrições de precedência do problema.

Portanto, para a resolução do problema, a partir da solução inicial selecionamos uma variável do tipo 0-1 e resolvemos dois problemas de programação linear, para valores fixos da variável, desenvolvendo uma arborescência. Este procedimento é repetido até encontrarmos uma solução que respeite as restrições do problema e seja, obviamente, ótima. Este processo de fixação de variáveis e resolução de problemas e conhecido como geração e avaliação de vértices ou estados.

É interessante observarmos que a fixação das variéveis 0-1 é equivalente ao processo decisório na solução de um conflito, ou seja:

Considerando, as variáveis 0-1 $x_{ki,j}$ e $x_{ik,j}$, para $x_{ki,j} = 1$,

teremos, consequentemente, $x_{ik,j} = 0$, e portanto:

$$t_{k,j} + p_{k,j} \leq t_{i,j} \quad e \quad t_{i,j} \leq t_{k,j} + M \,, \qquad (21)$$

caso contrário:

$$t_{i,j} + p_{i,j} \leq t_{k,j} \quad e \quad t_{k,j} \leq t_{i,j} + M. \qquad (22)$$

Um critério de poda comum para o procedimento "branch and bound" é estabelecermos limites superiores para o problema, sempre que for alcançada uma solução viável. Desta forma, sempre que o valor de um vértice for igual ou maior que um limite superior ("bound") estabelecido, podemos eliminá-lo do processo de ramificação ("branch").

A seguir, descreveremos alguns critérios utilizados no processo de ramificação ou escolha de vertices para expansão:

- Escolha de Vértices para Expansão:

A bem da verdade, sabemos que o algoritmo "branch and bound" é equivalente a um algoritmo de busca informada em Inteligência Artificial (IA). Desta forma, partindo-se de uma solução inicial (raiz) desenvolvemos uma árvore de busca em um grafo de estados (soluções parciais do problema) seguindo uma estratégia de ramificação (seleção de vértices para expansão) estabelecida. Os vértices ou estados a serem expandidos, podem ser colocados em uma lista a qual denominaremos lista de abertos.

Dentre as estratégias de ramificação principais, podemos citar:

1 - Escolha do estado na lista de aberto que apresente melhor valor, "best first". Desta forma, sempre que gerarmos novos estados, inserimos os mesmos na ordem crescente de valores (problema de minimização) na lista de abertos. A lista de abertos se manterá ordenada de modo que possamos retirar o primeiro da lista no processo de escolha. Caso o valor de algum estado gerado seja igual ou superior ao limite superior estabelecido para o problema, o mesmo pode ser excluído do processo de ramificação (não é inserido na lista de abertos). O algoritmo que adota esta estratégia de ramificação fornece uma solução ótima depois de um tempo finito de processamento.

2 - Escolha do último estado gerado para expansão, "depth first". Esta estratégia é equivalente a realização de uma busca em profundidade em um grafo de estados. Assim que é atingida uma solução viável, que não é, necessariamente, a ótima, é realizado um retorno "bactracking" ao último estado gerado. O algoritmo que adota esta estratégia alcanca rapidamente uma solução viável, entretanto a descoberta da solução ótima é mais demorada.

3 - Escolha do último estado gerado até atingirmos uma solução viável, ou o mesmo ser deletado do processo de ramificação. Em seguida, escolha do estado com menor valor. Neste caso, a descoberta de soluções viáveis são realizadas utilizando-se busca em profundidade e, em seguida, retornando ao estado da lista de abertos que apresenta menor valor, nao realizando, necessariamente, o "bactracking". O algoritmo que adota esta estratégia de ramificação pode ser considerado como um

algoritmo intermediário em relação aos dois primeiros.

- Fixação de Variáveis 0-1:

Considerando que um estado vai ser expandido, necessitamos determinar a variável 0-1 que será fixada. Se escolhermos uma variável qualquer, podemos não estar resolvendo um conflito, já que as restrições de integralidade são suficientes mas não necessárias a não ocorrência de conflitos.

Desta forma, é mais eficiente escolhermos a variévei do tipo 0-1 associada a um conflito qualquer do problema. Caso não haja conflito em um estado escolhido para expandir, a solução é considerada viável, mesmo se ainda houver ocorrência de variáveis 0-1 não fixadas.

O raciocínio acima é correto considerando que o número máximo de conflitos que podem ocorrer em um problema de sequenciamento de trens, é relativamente pequeno e pode ser determinado a priori .

Para determinarmos a ocorrência de um conflito basta identificarmos um trecho "j" tal que:

$$t_{k,j} + p_{k,j} \geq t_{i,j} \text{ se } t_{i,j} >/ t_{k,j} \qquad (23)$$

ou

$$t_{i,j} + p_{i,j} \geq t_{k,j} \text{ se } t_{k,j} >/ t_{i,j} \text{ .} \qquad (24)$$

Nos dois casos, o atendimento de algum trem só libera o trecho "j" , após a chegada do outro.

Portanto, se considerarmos o atendimento do trem "k" primeiro, devemos ter a inequação:

$$t_{i,j} \geq t_{k,j} + p_{k,j} \text{ , satisfeita.}$$

E, caso considerarmos o atendimento do trem "i" primeiro , a inequação a ser satisfeita deve ser:

$$t_{k,j} \geq t_{i,j} + p_{i,j}.$$

É interessante observarmos que a introdução de uma ou outra inequação ao conjunto de restrições do problema é equivalente a fixação das variaveis 0-1,já que as restrições em que aparecem a constante M são consideradas não ativas.

- Conflitos de Ultrapassagem :

Na resolução de um conflito de ultrapassagem [19..20], teremos duas possibilidades de ação. Considerando a parada do trem não prioritário , trem "k", no pátio anterior da seção "j" , teremos, que satisfazer as restrições :

$$t_{i,j} \geq t_{i,j} + p_{i,j} \qquad (25)$$

e

$$t_{k,j} \geq t_{i,j} + H_j \ , \qquad (26)$$

lembrando que a restrição [25] representa a restrição de sequenciamento do trem "i". Portanto, o trem "k" aguarda a liberação da seção "j" pelo trem "i".

Entretanto, se considerarmos a parada do trem "k" no pátio posterior da seção "j", teremos um atraso simultâneo dos trens "i" e "k" , determinando as seguintes restrições:

$$t_{i,j} + p_{i,j} \geq t_{k,j} + p_{k,j} + H_j$$

ou

$$t_{i,j+1} >/ p_{k,j+1} + H_j \qquad (27)$$

e

$$t_{k,j+1} >/ t_{i,j+1} + H_{j+1} \qquad (28)$$

3.3.4 - Método de Solução

Seguindo o raciocínio acima, podemos orientar a busca de uma solução ótima no grafo de estados do problema segundo a resolução de conflitos. Desta forma, a profundidade da árvore de busca será igual ao número de conflitos a serem resolvidos ate alcançarmos uma solução viável para o problema.

Outra observação importante, é que no problema relaxado, sem as restrições de integralidade, as restrições de precedência [17..18] são satisfeitas quanto a condição de disjunção na solução ótima do problema ,se considerarmos no desenvolvimento do mesmo as restrições do tipo [23] ou [24]. O mesmo é válido para as restrições de precedência que observam os conflitos de ultrapassagem, observando-se o conjunto de restrições [25..26] ou [27..28].

Portanto, podemos utilizar o seguinte algoritmo para a solução do problema de otimização de trens, seguindo a primeira estratégia de ramificação apresentada:

1 - Resolva o problema de sequenciamento de trens (algoritmo simplex), somente com as restrições de sequenciamento [10..12] , gerando a raiz da árvore de busca;

2 - Insira a raiz com o respectivo "tableau final" em uma lista de abertos;

3 - Selecione um estado da lista de abertos;

4 - Localize um conflito qualquer, e gere dois estados (filhos), considerando a introdução das restrições do tipo [23] ou [24] no "tableau" do estado antecessor (pai) se o conflito não for de ultrapassagem; caso contrário, introduza os conjuntos de restrições [25..26] ou [27..28] . Caso não haja conflito, a solução e ótima, fim;

5 - Calcule a função objetivo dos estados gerados (algoritmo dual do simplex), caso o valor seja maior ou igual ao limite superior estabelecido, delete o respectivo estado do processo de ramificação;

6 - Insira os estados gerados na lista de abertos segundo a ordem decrescente de valores;

7 - Retorne ao passo 3.

A idéia de acrescentarmos ao problema relaxado equações que garantam a solução de conflitos, pode ser considerada como a introdução de plano de cortes de modo a garantir a integralidade das variáveis de decisão.

3.3.5 - Restauração da Viabilidade:

Quando, no passo 4 do algoritmo, introduzimos uma nova equação

ao "tableau" do estado pai, tornamos o sistema primal inviável. Entretanto, a viabilidade do sistema pode ser obtida, utilizando o algoritmo dual do simplex (o sistema é dual viável).

Para a utilização do algoritmo dual do simplex é necessário guardarmos o "tableau final" do estado antecessor na forma completada do simplex (N. Maculan Filho [1978]) , que facilita a restauração da viabilidade do sistema por operações de colunas.

Para o estado inicial (raiz) do problema de sequenciamento de trens temos que satisfazer o seguinte sistema de restrições, segundo qualquer função objetivo clássica do problema de "scheduling":

$$t_{i,j} - s_{i,j} = r_i \text{ , p/ } \forall \text{ i e j} \qquad (29)$$

referentes a primeira seção, e

$$t_{i,j+1} - t_{i,j} - s_{i,j} = p_{i,j} \text{ , p/ } \forall \text{ i e j} \qquad (30)$$

referente as demais seções, e

$$t_{i,j} \text{ e } s_{i,j} \geq 0 \text{ , p/ } \forall \text{ i e j, i=1,...,n e j} \in R(i). \qquad (31)$$

Teremos, portanto, um sistema com nxp linhas e 2xnxp colunas , derivando uma base de trabalho de dimensões (nxp,nxp) e um "tableau" coluna sob a forma completada de dimensões (2xnxp+1,nxp+1), considerando o vetor de termos independentes e o vetor de multiplicadores.

A solução do sistema [29]..[31] é trivial, pois, tratando de um problema de minimização, basta tornarmos as variáveis de folga não-basicas e resolvermos o sistema acima por substituição de variáveis.

A solução inicial, equivalente a raiz, é um limite inferior para o problema de sequenciamento de trens, já que a ocorrência de atrasos devido a existência de conflitos e nula.

Se introduzirmos ao sistema [29]..[31], a equação do tipo:

$$t_{i,j} - t_{k,j} - s' = p_{k,j} \text{ ,} \qquad (32)$$

referente a um conflito existente no trecho "j",entre o trem "i"

e o trem "k", teremos que acrescentar uma nova linha ao "tableau" coluna da raiz do problema.

Para garantirmos a viabilidade do problema primal, teríamos a introdução da variável de folga $s_{i,j}$ e, consequentemente, a saída da variável de folga s' da base.

De fato, ao introduzirmos a equação [32], tornamos a restrição de sequenciamento correspondente não ativa (folgada), requerendo a retirada da variável de folga associada, do conjunto ativo do problema. Para tanto, é necessário que:

$$p_{k,j} + t_{k,j} \geq t_{i,j+1} - p_{i,j}, \quad \text{ou} \qquad (33)$$

$$p_{k,j} + t_{k,j} \geq t_{i,j},$$

que é a condição de ocorrência do conflito.

Neste caso, podemos admitir a retirada da restrição de sequenciamento do sistema, pois a mesma afeta somente o cálculo da variável $t_{i,j}$, cujo novo valor é estabelecido pela equação introduzida ao sistema, caso não seja necessária a sua utilização novamente. Desta forma, o "tableau" manterá sempre as mesmas dimensões.

Este procedimento será válido, como veremos adiante, se localizarmos os conflitos na ordem natural de ocorrência (estabelecendo uma hierarquia cronológica).

3.3.6 - Análise da Complexidade:

- Classe de Problemas:

Para provarmos que o problema de sequenciamento de trens é NP-completo, primeiramente, vamos analisar a complexidade do problema de "flow-shop scheduling" comparando o mesmo com um problema conhecido como "decision knapsack " segundo (M.M. Syslo, N. Deo e J.S. Kowalik [1983]). Em seguida, mostraremos que o problema de sequenciamento de trens, e um problema da classe NP, completando, assim, a nossa prova.

Prova:

O problema conhecido como " decision Knapsack " ou problema da mochila, consiste em:

" Dado um conjunto de números inteiros e positivos a1,a2,...,aq e b. Existe um subconjunto B, contido no conjunto acima, tal que:

$$\sum_{j \in B} a_j = b \quad ? \qquad (34)$$

O problema acima é NP-completo, portanto, se conseguirmos reduzi-lo, em tempo polinomial, a um problema de "flow-shop scheduling" semelhante, podemos determinar a complexidade do problema de "flow-shop".

De fato, se admitirmos, um problema de "flow-shop" com n = q+1 jobs e 3 máquinas, com tempos de processamento respectivamente iguais a:

$$(0,a_1,0),(0,a_2,0),\ldots,(0,a_q,0) \text{ e } (b,1,\sum_{j=1}^{q} a_j), \qquad (35)$$

o problema da mochila semelhante consiste em determinarmos se existe uma permutação de jobs com tempos de processamento somados iguais a :

$$\sum_{j=1}^{q} a_j + 1. \qquad (36)$$

Portanto, realizando a transformação inversa, também polinomial, concluímos que o problema de "flow-shop scheduling" é, também, um problema NP-completo.

O problema de "flow-shop" é idêntico a um problema de "job-shop" , acrescido das considerações:

. as "m" máquinas são ordenadas igualmente para todos os jobs, ou seja, é estabelecida uma ordem de processamento dos jobs relativo as "m" máquinas;

. nenhuma máquina pode processar mais de um job de cada vêz, e

. nenhum processamento de job pode ser dividido ou

interrompido.

Neste caso, como estas considerações tornam o problema de "flow-shop" mais simples e menos complexo que o problema de "job-shop", teremos o problema de sequenciamento de trens, também, Np-Completo . Ademais, estamos tratanto de um problema de "job-shop" dinâmico, com restrições de capacidade e, ao mesmo tempo, multi-critério .

- Complexidade do Grafo de Estados:

Sabemos que um grafo de estados é definido por um grafo direcionado consistindo de um conjunto de vértices (representativo dos estados) e arestas (representativo da ligação entre dois estados sucessivos). Cada vértice representa um estado do problema, existindo um vértice inicial ou raiz (identificado com o estado inicial) e um conjunto de vértices solução (identificado com os estados finais)

Podemos considerar as ligações entre dois estados sucessivos como o processo de geração de um estado a partir da aplicação de um operador ou uma função de transformação sobre o estado antecessor.

No processo de solução do problema, construímos uma árvore de busca binária, sobre o grafo de estados. Desta forma, partimos do nível inicial , correspondente a raiz da árvore, e prosseguimos na construção da mesma , até atingirmos um vértice solução .

Portanto, podemos, inicialmente, definir a complexidade do grafo de estados em função do espaço de estados do problema. Este espaço é normalmente determinado em função da aplicação dos operadores que determinam um fator de ramificação, e da distância que separa os estados finais do estado inicial do problema (nível de profundidade).

No problema de sequenciamento de trens, sabemos que cada estado quando expandido, produz dois novos estados sucessores, relativo as duas possíveis soluções de um conflito. Sabemos,também, que a profundidade de uma solução é determinada pelo numero de conflitos existentes.

Desta forma, podemos ter como possíveis soluções do problema 2^n estados finais, considerando "n" o numero de conflitos existentes.

O numero de conflitos em um problema de sequenciamento pode ser

facilmente determinado. Lembrando que :

$$\bar{F} = \bar{N} / n ,$$

e considerando o período médio de planejamento igual à 6 h (equivalente ao turno de trabalho do controlador). Teremos o valor de n igual a 3 (ou seja , a cada 1/3 h ocorre a chegada de um trem), e, ao longo de 6 h, o número provável de 18 trens em tráfego.

Considerando um balanceamento normal (9 trens em cada sentido) , teremos ao longo de 6 horas a possibilidade de ocorrência de 9x9 conflitos de cruzamento (pior das hipóteses). Portanto, sem considerarmos os conflitos de ultrapassagem e manutenção do "headway", teremos :

2^{81} possíveis soluções diferentes para este planejamento, caracterizando a explosão exponencial do problema.

De um modo geral, para "n" trens em tráfego,teremos um total de:

$$2^{(n/2)x(n/2)} \text{ cruzamentos, no caso pessimista.} \qquad (37)$$

Quanto aos conflitos de ultrapassagem ou manutenção do "headway" ,teremos , na pior das hipóteses, a existência de conflitos entre todos os trens de mesmo sentido. Desta forma, considerando, "n" trens em tráfego, teremos um total de:

$$2^{npx(np-1)/2} + 2^{nix(ni-1)/2} \text{ conflitos ,} \qquad (38)$$

onde "ni" se refere a quantidade de trens no sentido impar e "np" a quantidade de trens em sentido par (convenção de sentidos).

A divisão por 2 é devido ao fato de que a ocorrência deste conflito entre dois trens e mutuamente exclusiva, ou seja, se um trem "i" e ultrapassado pelo trem "k" na seção "j" , obviamente, o trem "k" não podera ultrapassar o trem "i" na mesma seção.

No exemplo anterior, teríamos, um total de:

2×2^{36} conflitos , considerando o balanceamento normal.

Portanto, considerando a profundidade da árvore de busca , dependente do número de conflitos, como uma função polinomial dos dados de entrada, com ordem máxima $O(n^2)$, caracterizamos o enquadramento do problema na classe de problemas NP (R.E. Campello e N. Maculan Filho [1989), completando, assim, a demonstração da seção anterior ■

- Exibição e Reconhecimento de uma Solução:

Se representarmos os tempos de ocupação de cada seção de um trecho ferroviário pelos respectivos trens em uma tabela de dimensões (nxp), onde "n" fosse o número de trens e "p" o número de seções sob planejamento, teríamos, para a identificação de conflitos, em uma solução exibida, os seguintes algoritmos:

. para trens de sentidos contrários:

```
Para j <-- 1 até p Faca
  Para i <-- 1 até  ni  Faca
    Para k <-- 1 até np Faca
      determine existência do conflito(i,k,j)
    FimPara
  FimPara
FimPara.
```

Neste caso, teríamos a função de complexidade da ordem máxima de O(ni.np.p).

. para trens de mesmo sentido:

```
Para j <-- 1 até p Faca
  Para i <-- 1 até ni Faca
    Para k <-- 1 até ni Faca
      Se i <> k Então
        determine existência de conflito(i,k,j)
      FimSe
    FimPara
  FimPara;
  Para i <-- 1 até np Faca
    Para k <-- 1 até np Faca
      Se i <> k Então
        determine existência de conflito(i,k,j)
      FimSe
    FimPara
  FimPara
FimPara.
```

Neste caso, teríamos a função de complexidade da ordem máxima de O(max{p.ni.(ni-1),p.np.(np-1)}).

Desta forma, consideramos, também, a existência de um algoritmo polinomial para o reconhecimento da justificativa do problema de decisão associado ao problema de sequenciamento de trens, ratificando, assim, sua pertinência a classe NP (R.E. Campello e N. Maculan Filho [1989]).

A complexidade da árvore de busca esta relacionada, como vimos, ao espaço de estados do problema e também, principalmente, aos critérios de poda e a função heurística utilizada para a realização de uma busca informada.

Tratando-se de um problema de grande complexidade, tentaremos na medida do possível, reduzir o espaço de estados do problema. Para tanto, sera necessário o desenvolvimento de uma boa função heurística , a obtenção de bons limites e critérios de poda e, também, a utilização de um sistema de produção que oriente, seletivamente, a geração de novos estados, reduzindo o fator de ramicação do grafo de estados e , consequentemente, o numero de estados finais.

3.4 - Otimização "Fuzzy"

3.4.1 - Introdução

Nesta seção, apresentaremos uma breve descrição da teoria sobre otimização "fuzzy" necessária ao desenvolvimento de uma formulação multi-objetiva " fuzzy " para o problema de sequenciamento de trens, bem como para modelagem da imprecisão relativa aos paramêtros existentes, procurando, desta forma, retratar o caráter impreciso, dinâmico e flexível do problema.

A utilização da teoria de "fuzzy sets" em problemas de otimização tem crescido muito ultimamente . Convém mencionarmos,a aplicação da teoria em problemas de otimização multi-objetiva e programação objetiva ("goal programming"), ressaltando os trabalhos de (H.-J. Zimmermann [1978]), (B. Werners [1987]), (E.L. Hannan [1981]) e (P.A. Rubin, R. Narasimhan [1984]) para problemas com funções

lineares e (M. Sakawa, H. Yano [1986]) para problemas com funções não lineares.

Destacam-se, também, os trabalhos relativos ao desenvolvimento de operadores para retratarem a expressão lógica "and" que define o conjunto de decisões "fuzzy". Neste caso, podemos ressaltar o trabalho de (M.K. Luhandjula [1982]) sobre a utilização de operadores compensatórios, o trabalho de (U. Thole, H.-J. Zimmermann e P. Zysno [1979]) onde apresentam um estudo sobre a aplicabilidade dos operadores "min" e "produto" para expremirem a interseção de conjuntos "fuzzy" e ,finalmente, o trabalho de (H-j. Zimmermann e P. Zysno [1980]) onde é realizado um estudo sobre a utilização de compensação envolvendo união e interseção de conjuntos "fuzzy".

No que se refere a existência de "fuzziness" nos coeficientes das restrições ou no objetivo do problema, podemos destacar o trabalho de (H . Tanaka, K. Asai [1984]) sobre parâmetros e funções "fuzzy", (J.L.Verdegay [1984]) que apresenta um estudo sobre dualidade em programação "fuzzy" e o trabalho de (D. Dubois , H. Prade [1980]) que apresenta um estudo sobre restrições "fuzzy" com números "fuzzy".

Para o nosso estudo, tratando-se de um problema com objetivos "fuzzy", será importante a análise dos trabalhos de Zimmermann [1978], Werners [1987], Hannan[1981] e Rubin, Narasimhan[1984]. Quanto a existência de parametros "fuzzy", demos importância aos trabalhos de Dubois,Prade[1980], Tanaka, Asay[1984] e Verdegay[1984].

Convém, ainda, ressaltarmos os trabalhos de (H. Prade[1979]) e (R. Fonseca Neto[1988]) que apresentam aplicações da teoria de "fuzzy sets" em problemas de "scheduling" com parâmetros "fuzzy".

3.4.2 - Programação Flexível

Em seu trabalho sobre otimização "fuzzy", C.V. Negoita [1981] propõe uma interpretação da programação "fuzzy" como uma programação flexível. Neste caso, otimalidade significa efetividade, contrariando a noção de eficiência da otimização convencional.

Definindo a qualidade de uma alternativa em um conjunto de

possíveis soluções, em função do grau de satisfação das restrições e objetivos do problema, tomados para determinados intervalos que caracterizam regiões de viabilidade . O problema se torna determinar uma alternativa tal que os "fuzzy sets" associados ,que expressam uma relação de preferência nestes intervalos, sejam os maiores possíveis. Neste sentido, restrições e objetivos são vistos como "fuzzy sets" do conjunto de alternativas.

- Soluções Satisfatórias:

O problema clássico de otimização multi-objetiva pode ser colocado na forma:

$$\text{Min } H(x) \tag{39}$$

sujeito a:

$$g_i(x) \leq b_i \ , \ i=1,2,\ldots,m \tag{40}$$

$$e \quad x \geq 0 \ , \tag{41}$$

onde $H(x) = (h_i(x), i=1,\ldots,k)$ é a função multi-objetiva do problema.

"Fuzzy sets" são conjuntos definidos por funções de pertinência que indicam o grau de satisfação associado as equações do sistema [39..41], incluindo os níveis de aspiração desejados. Desta forma , se alguma equação do sistema é fortemente violada, temos um valor para a função de pertinência igual a zero. Se a equação é satisfeita, temos um valor para a função de pertinência igual a 1 e,finalmente, se a equação é violada dentro de um limite razoável, a função de pertinência pode tomar valores entre 0 e 1 .

Um ponto x é considerado uma solução eficiente do problema [39..41], se não existe nenhum x' viável, tal que:

$$h_i(x') \leq h_i(x), \text{ p/ } i=1,\ldots,k \ , \quad e \tag{42}$$

$$h_i(x') < h_i(x), \text{ p/ no mínimo um } i \in \{1,\ldots,k\} \ . \tag{43}$$

O conjunto de pontos eficientes, também conhecidos como não dominados, e denominado conjunto ótimo Pareto.

Podemos considerar que uma solução satisfatória faz parte do conjunto de soluções eficientes do problema multi-objetivo. De fato, uma solução não eficiente não é atrativa para o tomador de decisão, já que é possível acharmos uma solução no mínimo tão boa quanto todos os objetivos e superior em ao menos um.

Se considerarmos intervalos de satisfação para cada restrição , podemos colocar o problema [39..41] como determinar uma efetiva escolha, ou seja, determinar um ponto $x \geq 0$, tal que:

$$wc_i \leq h_i(x) \leq Wc_i \quad p/ \quad i=1,2,\ldots,k, \quad e \tag{44}$$

$$wb_i \leq g_i(x) \leq Wb_i \quad p/ \quad i=1,2,\ldots,m \ . \tag{45}$$

É importante observarmos que a função objetiva é tratada como uma restrição qualquer, quando associamos níveis de aspirações aos objetivos do problema. Temos, portanto, um sistema de inequações que permite a escolha de boas soluções.

Os valores wb_i, wc_i e Wb_i, Wc_i se referem ao limite inferior e superior de seus respectivos intervalos relativos as restrições e aos objetivos do problema.

Este sistema de objetivos e restrições flexíveis, também foi descrito por (B. Werners [1987), que desenvolve, igualmente, a idéia de um conjunto de soluções "fuzzy" eficiente, generalizando a definição de eficiência da programação multi-objetiva clássica.

- Funções de Pertinência:

Zimmermann [1978], estabelece uma função linear para medir os valores de pertinência . Considerando uma versão "fuzzy" do sistema [39..41] e tornando seus respectivos intervalos abertos inferiormente, teremos:

$$h_i(x) \lesssim c_i, \quad i=1,2,\ldots,k \ , \ e \tag{46}$$

$$h_i(x) \lesssim b_i \ , \quad i=1,2,\ldots,m \tag{47}$$

$$e \quad x \geq 0 \ . \tag{48}$$

Desta forma, Zimmermann define funções de pertinencia "μ_j", tal que:

$$\mu_j(h_i(x),g_i(x)) = 0 , \quad se\ (h_i(x),g_i(x))_j > (Wc_i,Wb_i) ,$$

$$0 < \mu_j(h_i(x),g_i(x)) < 1 , se\ (c_i,b_i) < (h_i(x),g_i(x))_j \leq (Wc_i,Wb_i),$$

ou seja, podemos determinar μ_j calculando:

$$\mu_j(h_i(x),g_i(x)) = 1 - ((h_i(x),g_i(x)) - (c_i,b_i) / d_i, \quad (49)$$

onde: $d_i = (Wc_i,Wb_i) - (c_i,b_i)$ e, finalmente,

$$\mu_j(h_i(x),g_i(x)) = 1 , \quad se\ (h_i(x),g_i(x))_j \leq (c_i,b_i).$$

Os valores "d_i" se referem aos limites aceitáveis correspondente a violação das restrições ou dos objetivos do problema.

Os valores "c_i" podem ser considerados como o ótimo individual de cada objetivo do problema, e os valores pessimistas "Wc_i" são determinados, da seguinte forma:

Suponha que h_i^j represente o valor do i-ésimo objetivo, quando substituimos os valores das variáveis de decisão, pela solução do problema:

$$Min\ h_j(x) \quad (50)$$

$$sujeito\ a:\ g_i(x) \leq b_i,\ i=1,2,\ldots,m \quad (51)$$

$$e\ \ x \geq 0 \ , \quad (52)$$

então:

$$Wc_i = Min_j \{ h_j^i ,\ j=1,\ldots,k \}\ \ p/\ i=1,\ldots,k\ . \quad (53)$$

- <u>Operadores Clássicos</u>:

A decisão em um sistema "fuzzy" é vista como a interseção de equações ("sets") "fuzzy". Na teoria "fuzzy sets" a interseção de conjuntos, normalmente, corresponde a expressão logica "and", que pode ser representada pelo operador "min" ou pelo operador produto segundo R.E. Bellman e L.A. Zadeh [1970].

Portanto, a fução de pertinência do conjunto solução, utilizando

o operador "min", será:

$$\mu_m(h_i(x),\ g_i(x))_m = \text{Min}_j\ \mu_j(h_i(x),g_i(x)). \qquad (54)$$

Se considerarmos a utilização do operador produto teremos:

$$\mu_m(h_i(x),\ g_i(x))_m = \Pi_j\ \mu_j(h_i(x),g_i(x)). \qquad (55)$$

- Formulações:

Podemos, também, definir a solução com mais alto grau de pertinência, como a solução maximizante, desta forma, devemos determinar:

$$\text{Max}_{x \geq 0}\ \text{Min}_j\ \mu_j(h_i(x),g_i(x)), \qquad (M1)$$

ou

$$\text{Max}\ z$$

$$z \leq \mu_i(h_i(x)),\ i=1,2,\ldots,k \quad \text{e}$$

$$z \leq \mu_i(g_i(x)),\ i=1,2,\ldots,m,$$

$$\text{e}\ \ x \geq 0.$$

Ou, utilizando o operador produto:

$$\text{Max}_{x >/ 0}\ \Pi_j\ \mu_j(h_i(x),g_i(x)). \qquad (P1)$$

A utilização do operador "max" pode ser considerado como uma forma de obtermos uma solução preferida entre um conjunto de soluções não dominadas (L.F.B. Baptistella, A. Ollero [1980]), ou seja, escolhemos a solução com mais alto grau de pertinência relativa ao conjunto solução, que é a interseção de conjuntos "fuzzy" associados aos objetivos e restrições do problema (B. Werners[1987]).

- Soluções Não Dominadas:

J.J. Buckley [1983] apresenta um estudo estabelecendo condições

para que a solução de um problema de programação "fuzzy" seja uma solução não dominada ou um ponto do conjunto ótimo de Pareto relativo ao problema [39..41].

Segundo Buckley, se considerarmos a utilização dos operadores produto e min, as funções de pertinência associadas aos objetivos do problema devem ser monotonicamente decrescentes, podendo ser lineares, não lineares ou lineares por partes. Portanto, a utilização de funções de pertinência com forma triangular ou de uma distribuição normal pode estabelecer uma solução dominada ou não pertencente ao conjunto de soluções eficientes.

A demonstração de Buckley é baseada na idéia de que a resolução dos problemas de programação "fuzzy" M1 e P1 são equivalentes a resolvermos:

$$\text{Max}\{ G_m(v) \mid v \in V \}, \text{ onde } G_m(v) = \text{Min}_j \{ \mu_j(v_j) \}, \qquad (M2)$$

ou

$$\text{Max}\{ G_p(v) \mid v \in V \}, \text{ onde } G_p(v) = \Pi_j \{ \mu_j(v_j) \}, \qquad (P2)$$

$$\text{onde: } V = F(x) = (h_i(x), g_i(x)), \qquad (56)$$

de tal forma que, se algum $v' \in V$, que resolve M2 ou P2, for um ponto eficiente; então, pelo menos um ponto $x' \in F^{-1}(v')$ que resolve M1 ou P1, será um ponto não dominado.

Prova:

Vamos analizar a demonstração de Buckley, considerando que v resolve o problema M2, que o conjunto V seja convexo,e que F(x) = H(x).

Para que x não seja um ponto eficiente e,portanto, dominado, é necessário que exista um ponto x' associado a função H(.) do problema de programação multi-objetiva ,tal que:

$$h_i(x') \leq h_i(x) \quad p/ \quad i = 1,\ldots,k \quad e$$

$$h_j(x') < h_j(x) \quad p/ \text{ algum } j \in \{1,\ldots,k\}.$$

Mas, se $h_j(x') < h_j(x)$, consequentemente,

$$v \leq \mu_j(h_j(x)) < \mu_j(h_j(x')),$$

e

$v \leq \mu_i(h_i(x'))$ para $i = 1,\ldots,m$,

dado ao fato de $\mu(.)$ ser monotonicamente decrescente e contínua.

Considerando a função inversa de $\mu_j(.)$, ou seja:

se $\mu_j(h_j(x)) \geq v \Rightarrow h_j(x) \geq \mu_j^{-1}(v)$,

e a existência de v' associado a F(x') ou a H(x') , teremos:

$\mu_j^{-1}(v) \leq h_j(x) < h_j(x')$ e

$\mu_i^{-1}(v) \leq h_i(x')$ p/ $i= 1,\ldots,m$.

Portanto:

$\mu_j^{-1}(v) < \mu_j^{-1}(v')$, e

$\mu_i^{-1}(v) \leq \mu_i^{-1}(v')$ p/ $i=1,\ldots,m$.

Como $\mu_i(.)$ é monotonicamente decrescente, temos que $v < v'$, ou seja, concluímos que v não faz parte da solução ótima do problema M2.

Segundo B. Werners [1987], podemos afirmar, que neste caso, v não e um ponto "fuzzy" eficiente, sendo dominado por v'.

Entretanto, caso v resolva M2, e considerando V convexo, não conseguimos determinar nenhum ponto v', tal que:

$\mu_j^{-1}(v) < \mu_j^{-1}(v')$ p/ algum $j \in \{1,\ldots,m\}$ e

$\mu_i^{-1}(v) \leq \mu_i^{-1}(v')$ p/ $i=1,\ldots,m$,

e, consequentemente, o ponto x associado a $H^{-1}(v)$ é um ponto não dominado ■

Segundo Werners e Buckley, se (z^*,x^*) é uma solução ótima do problema M1 ou P1, entao x^* é uma solução não dominada do problema de programação multi-objetiva [39..41].Entretanto, caso o problema M1 ou P1 tenham várias soluções ótimas, e constatado que ao menos uma garante uma

solução não dominada. Buckley [1983] apresenta esta demonstração, considerando, ainda, a relaxação da condição de convexidade imposta ao conjunto V.

3.4.3 - Programação Robusta:

- Função "Fuzzy":

Seja o problema clássico de otimização , com objetivo fuzzy, seguindo a notação de J.L. Verdegay [1982], podemos escrever:

$$\tilde{\text{Min}} \quad h(x) \tag{57}$$

$$\text{sujeito a: } g_i(x) \lesssim b_i, \; i=1,2,\ldots,m \tag{58}$$

$$\text{e} \quad x \geq 0 \;, \tag{59}$$

onde o símbolo ~ implica a existência de "fuzziness" nos parâmetros que compõe as restrições e a função objetiva do problema.

O princípio de "fuzziness" de J.A. Goguen[1967] assume a existência de uma função de pertinência associada ao objetivo "fuzzy", já que o mesmo é definido como um conjunto "fuzzy" do conjunto de objetivos.

Neste caso, segundo C.V Negoita[1985], estamos tratando de um tipo de programação denominada programação robusta, que admite a existência de "fuziness" em seus parâmetros.

Seja a função h(x,a) relativa a um problema de otimização e a existência de dois conjuntos X e Y. Se o parâmetro "a" é dado por um número "fuzzy" $\tilde{A}$, devemos substituir a função h(x,a) por uma função "fuzzy" denominada $h(x,\tilde{A})$. Neste caso, quando um valor de x for dado, teremos um mapeamento "fuzzy" na forma:

$$h : X \rightarrow \mathcal{F}(Y), \quad \tilde{Y} = h(x,\tilde{A}), \tag{60}$$

onde: $\mathcal{F}(Y)$ é o conjunto de todos sub-conjuntos "fuzzy" de Y.

Portanto, podemos considerar o conjunto "fuzzy", $\tilde{Y} = h(x,\tilde{A})$,

associado aos valores do numero "fuzzy" $\tilde{A}$, seguindo o princípio da extensão (C.V. Negoita, D.A. Ralescu [1975]) que é uma extensão natural do conceito de mapeamento de conjuntos. Ou seja, dado valores de "x", podemos obter o mapeamento do conjunto "fuzzy" $\tilde{Y}$ a partir do conjunto "fuzzy" $\tilde{A}$.

Concordando com o princípio de "fuzziness" de Goguen, H. Tanaka e K. Asai [1984] definem o conjunto "fuzzy" $\tilde{Y}$, a partir de uma função de pertinência, dada por:

$$\mu_Y(y) = \begin{cases} \underset{\langle a \mid y=h(x,a)\rangle}{\text{Max}} \left[\mu_A(a) \right], \{a \mid y=h(x,a)\} \neq 0, \\ 0, \text{ caso contrário,} \end{cases} \qquad (61)$$

onde $\tilde{A}$ e um conjunto "fuzzy" relativo ao produto $[a_1 \times a_2 \times \ldots a_n]$ cuja função de pertinencia é dada por $\mu_A(a)$.

- Parâmetros "Fuzzy":

Tanaka e Asai [1984] limitam a definição de parâmetros "fuzzy" a um tipo de conjuntos "fuzzy", cuja função de pertinência assume a forma linear. Mais precisamente, os parâmetros "fuzzy" são definidos por conjuntos "fuzzy" representados por:

$$\mu_A(a) = \text{Min}_j \left[\mu_{Ai}(a_j) \right] , \qquad (62)$$

onde:

$$\mu_{Ai}(a_j) = \begin{cases} 1 - |\alpha_j - a_j| / c_j \quad p/ \quad \alpha_i - c_j \leq a_j \leq \alpha_i + c_j \ , \\ 0, \text{ caso contrário,} \end{cases} \qquad (63)$$

onde $c_j > 0$.

Desta forma, consideramos a existência de funções de pertinência:

$$\mu_{Ai}(a_j) : R \rightarrow [0,1], \qquad (64)$$

relacionadas a cada parâmetro (custo) componentes da função objetiva. A utilização do operador "min" é uma forma natural de

garantir o comprimento da satisfação mínima.

De acordo com a figura 29,podemos atribuir um significado físico a definição de Tanaka e Asai para um parâmetro "fuzzy" , concordando com a expressão " aproximadamente α " para uma determinada largura "c".Portanto, caso o parâmetro "a" assuma o valor central "α" , teremos o valor da função de pertinência igual a 1. Para valores de "a" além do intervalo de variação permitido, teremos o valor da função de pertinência igual a 0.

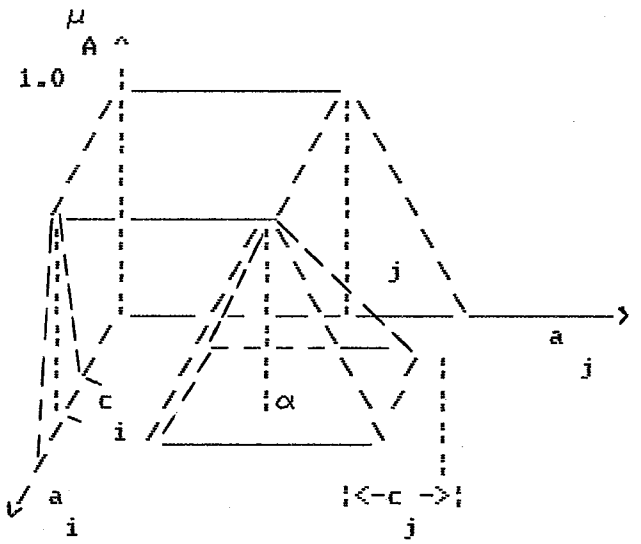


figura 29:"fuzzy set" relativo ao parâmetro "aproximadamente α".

Dado o parâmetro fuzzy $\tilde{A} = \{\alpha, c\}$, Tanaka e Asai definem a função linear "fuzzy":

$$\tilde{Y} = \tilde{A}_1 x_1 + \tilde{A}_2 x_2 + \ldots + \tilde{A}_n x_n = \tilde{A}\, x \quad , \qquad (65)$$

onde: $(\tilde{A}_1, \ldots, \tilde{A}_n)$, $(x_1, \ldots, x_n)$ são componentes dos vetores $\tilde{A}$ e x. Esta função é calculada a partir da função de pertinencia:

$$\mu_Y(y) = \begin{cases} 1 - |y - x^T| \,/\, c^T|x| \;, & x \neq 0 \;, \\ 1, \quad \text{se } x = 0 \quad \text{e} \quad y = 0, \\ 0, \quad \text{se } x = 0 \quad \text{e} \quad y \neq 0, \end{cases} \qquad (66)$$

onde: $|x| = (|x|_1, \ldots, |x|_n)$ e $\mu_Y(y) = 0$ se $c^T|x| \leq |y - x^T\alpha|$.

A função de pertinência obtida acima deriva, normalmente, da

aplicação das formulas [61..63]. A demonstração da prova deste resultado se encontra no trabalho de Tanaka e Asai.

Para melhor interpretarmos o significado de [66] , vamos admitir, como exemplo, a seguinte função linear "fuzzy":

$$Y = \tilde{A}_1 x + \tilde{A}_2 x \ ,$$

onde $\tilde{A} = \{(2,3),(1,2)\}$. Se $x = (1,1)^T$, o conjunto fuzzy $\tilde{Y} = (5,3)$ obtido significa que o conjunto "fuzzy" "aproximadamente 2" mais " aproximadamente 3" e igual a " aproximadamente 5 ".

Podemos considerar também a idéia de números "fuzzy", desenvolvida por Dubois e Prade[1978], onde um número fuzzy m do tipo LR é representado por três parâmetros (m,a,b), tal que:

$$\mu_m(x) = \begin{cases} L\ ((m - x) / a) & \text{para } x \leq m \ , \\ R\ ((x - m) / b) & \text{para } x \geq m \ , \end{cases} \qquad (67)$$

onde:

L e R são funções simétricas e L(0) e R(0) = 1 , o parâmetro "m" corresponde a um valor central e os parâmetros "a" e "b" correspondem a desvios aceitáveis. Quando os valores de "a" e "b" forem iguais a zero, "m" sera considerado um número exato.

Pelo princípio da extensão podemos determinar a função de pertinência da soma de dois números "fuzzy" m e n do tipo LR através da expressão :

$$u_{m+n}(z) = \text{Max}_{z=x+y} \text{Min} \{ u_m(x) , u_n(x) \} \ , \qquad (68)$$

derivando:

$$(m,a,b)_{LR} + (n,c,d)_{LR} = (m+n,a+c,b+d)_{LR} \ . \qquad (69)$$

- Dualidade:

Verdegay [1982], apresenta uma formulação alternativa para o problema de programação matemática com "fuzziness" no objetivo.

Seja [64] as funções de pertinência associadas aos parâmetros da função objetiva , neste caso é necessário resolvermos o problema:

$$\tilde{\text{Min}} \quad h(x,a) \tag{70}$$

sujeito a:

$$\mu_A(a) \leq 1 - h \ , \tag{71}$$

$$h_i(x) \leq b_i \ , \ i=1,2,\ldots,m \ , \tag{72}$$

$$e \quad x \in X \ , \ h \in [0,1], \tag{73}$$

onde $\mu_A(a)$ é definido por [62].

Como o problema acima apresenta solução difícil, Verdegay apresenta uma formulação alternativa baseada na propriedade de monotonicidade e continuidade das funções de pertinência. Neste caso, para cada função $\mu_{Aj}(a_j)$, $j=1,\ldots,n$, devemos ter:

$$\text{Se } \mu_A(a) \leq 1-h \implies \mu_{Aj}(a_j) \leq 1-h \implies a_j \leq \mu_{Aj}^{-1}(1-h), \tag{74}$$

considerando as propriedades de continuidade e monotonicidade das funções de pertinência.

Substituindo [74] em [71], temos:

$$\tilde{\text{Min}} \quad h(x,a) \tag{75}$$

sujeito a:

$$a_j \leq \mu_{Aj}^{-1}(1-h) \ , \ j=1,\ldots,n \ , \tag{76}$$

$$h_i(x) \leq b_i \ , \ i=1,2,\ldots,m \ , \tag{77}$$

$$e \quad x \in X \ , \ h \in [0,1]. \tag{78}$$

Mas, este problema é equivalente a:

$$\text{Min} \quad \sum_{j=1}^{n} \bar{\mu}_{Aj}^{-1}(1-h).x_j \tag{79}$$

sujeito a:

$$h_i(x) \leq b_i \ , \ i=1,2,\ldots,m \ , \tag{80}$$

$$e \quad x \in X \ , \ h \in [0,1], \tag{81}$$

considerando que a função $h(x,a) = \sum_{j=1}^{n} a_j . x_j$.

O problema de programação "fuzzy" acima é um problema de programação linear paramêtrica.

Neste mesmo trabalho, Verdegay mostra que dado um problema de programação "fuzzy", é possível resolver o seu dual fornecendo a mesma solução "fuzzy". Mostra, também, que se o problema primal apresenta "fuzziness" nas restrições, o problema dual é resolvido como um problema com "fuzziness" no objetivo e vice-versa.

3.5 - Desenvolvimento de um Modelo Fuzzy

3.5.1 - Avaliação dos Objetivos

- Avaliação do Desempenho:

Na seção 3.2.4, referente a descrição do problema de sequenciamento de trens, quando apresentamos um estudo sobre funções objetivas clássicas do problema de "scheduling", verificamos a possibilidade de minimizarmos o tempo total de espera no sistema , minimizando o tempo médio de término da última operação, o que é equivalente a :

$$\text{Min} \quad \sum_{i=1}^{n} t_{i,j} \quad , \tag{82}$$

onde: "j" se refere a ultima seção da rota R(i), estabelecida para cada trem "i", i=1,...,n .

Considerando que a variável $t_{i,j}$ representa uma medida do desempenho de cada trem "i" ao longo de sua rota R(i), podemos admitir a existência de um conjunto de objetivos, cada qual, associado ao valor da mesma, tal que:

$$\mu_i (x) = f (t_{i,j}) \quad , \tag{83}$$

ou seja , a função de pertinência que estabelece o nível de satisfação de cada objetivo (relativo a um trem ou a um grupo de trens) é uma função do instante de ocupação da última seção relacionada a respectiva rota.

No nosso trabalho, vamos considerar o desenvolvimento de uma função linear por partes, côncava, associada a cada objetivo do problema , estabelecendo uma hierarquia de objetivos flexivel amarrada a um conjunto de prioridades. O desenvolvimento desta classe de funções foi proposto, inicialmente, por P.A. Rubin e R. Narasinham [1984].

É importante ressaltarmos, que o cumprimento de um nível mínimo

aceitável de pertinência é de maior importância, estabelecendo um certo caráter compensatório no restabelecimento de prioridades.

- Níveis de Satisfação e de Prioridades:

Segundo Rubin e Narasimham[1984], é razoável considerarmos que a satisfação marginal do tomador de decisões decresce a medida que o nível de satisfação (grau de pertinência) relacionado ao objetivo aumenta.

Inicialmente, consideramos como o tempo de percurso ideal de cada trem, ou o cumprimento máximo de um objetivo, a solução ótima individual, c_i, ou seja, o tempo de início de ocupação da última seção, considerando a não ocorrência de atrasos.

Em seguida, consideramos como o atraso máximo (desvio) a hipótese do trem parar em todos os conflitos e aguardar o tempo máximo relativo a liberação da seção disputada.

Desta forma, ver quadro 4, podemos definir níveis de prioridade, com os respectivos valores dos níveis máximos de satisfação associados aos respectivos intervalos, ou seja, calculamos para cada nível de prioridade "k" a maxima satisfação "B_k" em uma escala de 0 a 1, tal que:

$$1 \geq B_1 \geq B_2 \geq ,\ldots, \geq B_n \geq 0 . \qquad (84)$$

quadro 4 : níveis de satisfação e prioridades:

Prioridade	Nível(k)	B_k	Intervalo	u_k
Pouco Imp.	1	1.0	0 - 10 % atraso	c_i
Meio Imp.	2	0.9	10 - 20% atraso	c_i + 0.1.atraso
Importante	3	0.7875	20 - 30% atraso	c_i + 0.2.atraso
Muito Imp.	4	0.6565	30 - 40% atraso	c_i + 0.3.atraso
Crítica	5	0.4522	40 - 50% atraso	c_i + 0.4.atraso
Inaceitável	6	0.0	> 50% atraso	c_i + 0.5.atraso

Os valores obtidos para o vetor B, estão relacionados aos coeficientes angulares dos segmentos de retas que formam a função linear por partes.

3.5.2 - Desenvolvimento das Funções de Pertinência

- Expressão Matemática:

Considerando o intervalo e o limite superior de cada nível de prioridade, construimos uma aproximação linear por partes da função de pertinência de cada objetivo "i" na forma:

$$\mu_i(t_{i,j}) = B_{k+1} + \frac{B_k - B_{k+1}}{u_{k+1} - u_k} \cdot (u_{k+1} - t_{i,j}), \qquad (85)$$

para $u_k \leq t_{i,j} \leq u_{k+1}$ e $k=1,\ldots,5$.

É interessante observarmos que, para cada objetivo do problema:

$\mu_i(u_k) = B_k$, p/ $k=2,\ldots,5$ e

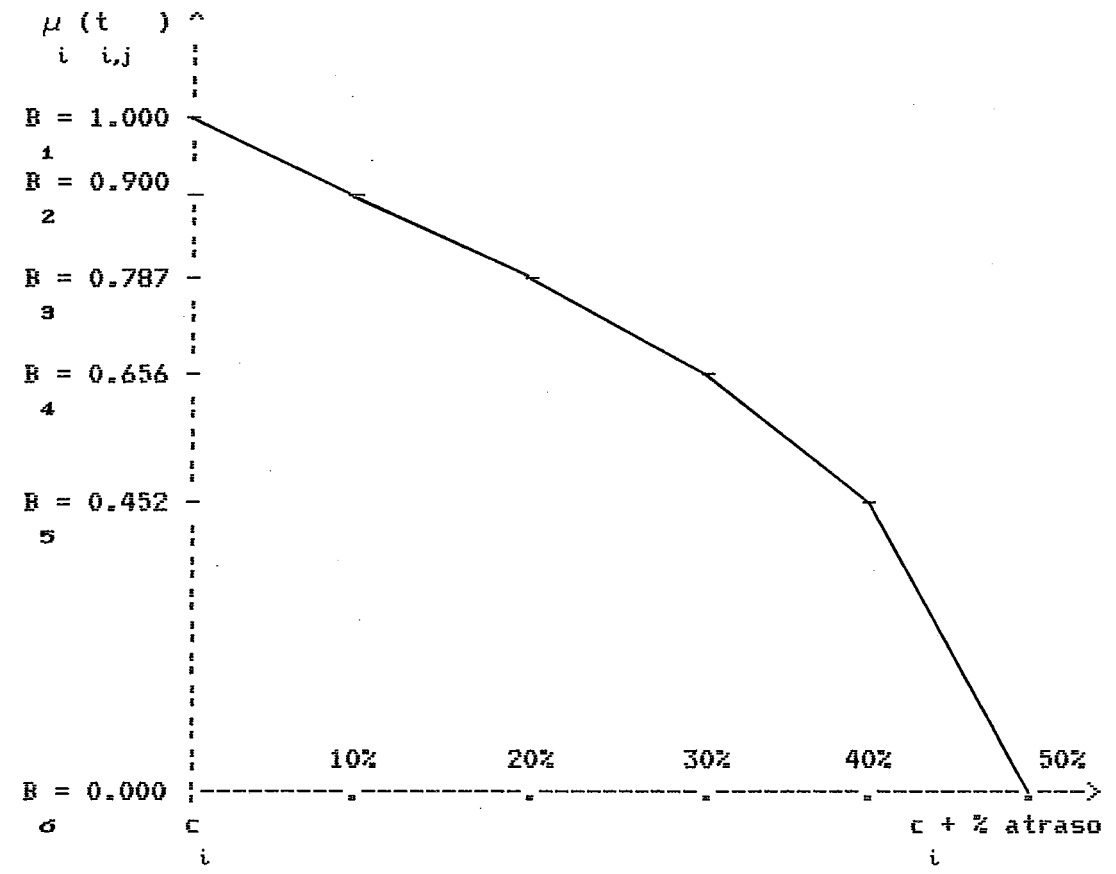
$\mu_i(c_i) = 1$, p/ $k=1$, correspondendo ao máximo valor individual.

A função de pertinência, definida em [85], tem a forma apresentada na figura 30.

- Propriedades:

É importante repararmos que o nível de satisfação do tomador de decisões $\mu_i(t_{i,j})$ não cresce tão rapidamente próximo do objetivo ideal (c_i), caracterizando, desta forma, o fenômeno de saturação que é exibido pelas funções de pertinência não lineares e não totalmente côncavas (P.A. Rubin,R. Narasimhan [1984]).

A função de pertinência definida em [85] é monotonicamente decrescente, garantindo a geração de pontos eficientes quando consideramos a resolução do problema de programação multi-objetiva "fuzzy" .



Nível	Prioridade
1	pouco importante
2	meio importante
3	importante
4	muito importante
5	crítico
6	inaceitável

figura 30: função de pertinência e níveis de prioridade.

A condição de concavidade é garantida pelo fato de que:

$$(B_k - B_{k+1} \,/\, u_k - u_{k+1}) \leq (B_{k-1} - B_k \,/\, u_{k-1} - u_k) \text{ , p/ } k=2,\ldots,5 \text{ .} \quad (86)$$

3.5.3 - Determinação dos Intervalos (Atrasos)

- Agrupamento de Objetivos e Cálculo de Atrasos:

A definição de 5 niveis de prioridades, estabeleceu a divisão do tempo de atraso em 5 intervalos considerados aceitáveis. Desta forma, não é permitido que um trem ou grupo de trens tenham um atraso maior ou igual a 50% do atraso máximo calculado.

A determinação do atraso máximo de cada trem ou grupo de trens (neste caso, estamos promovendo um agrupamento de objetivos) é de fundamental importância no desenvolvimento das funções de pertinência .

Para a estimativa do atraso, vamos considerar, que não haja trens prioritários na ferrovia. Neste caso,o atraso máximo será igual ao número previsto de conflitos x tempo máximo de espera por conflito.

Portanto, considerando o cálculo do atraso máximo, um objetivo aceitável seria ocorrer um atraso correspondente a 25% do atraso máximo para cada trem ou grupo de trens . Este atraso e consequência normal do movimento de trens, considerando a preferência no cruzamento para aquele trem que estiver mais próximo a seção na qual ocorrerá o conflito.

Consideramos, da mesma forma, um objetivo inaceitável se ocorrer um atraso superior a 50% do tempo de percurso teórico.

Podemos justificar este raciocínio baseado no fato de que um trem, mesmo parando em todos os conflitos, tem um atraso esperado em torno de 50% do atraso máximo, considerando a hipótese mais favorável e a menos favorável de tempo de espera do mesmo. Entretanto, se estabelecermos uma probabilidade de 50% relativa a hipótese de parada do trem (conflito entre trens de mesma prioridade), teremos um atraso normal em torno de 25% do atraso máximo.

Neste caso, o tempo máximo de espera por conflito é considerado como o tempo máximo necessário para ocorrer a liberação da seção disputada.

No caso da existência de trens prioritários, os mesmos terão preferência automática nos cruzamentos,com os trens de menor prioridade.

Portanto, estes trens terão um atraso normal inferior a 25% do atraso máximo, enquanto que, os trens de baixa prioridade, terão um

atraso normal superior a este percentual.

- Estabelecimento de Critérios:

Para a determinação dos intervalos associados a cada objetivo do problema, podemos admitir , também, a existência de critérios que estabelecam um limite superior nos tempos de percurso de um trem ou grupo de trens na ferrovia, obedecendo a estratégias superiores que orientam o gerenciamento e controle de trens e, consequentemente, a exucação dos planos de rotas. Como exemplo, podemos considerar os seguintes critérios:

- O trem de passageiros não deve ter um atraso final superior a 10% do tempo previsto de percurso. Neste caso, devemos considerar o atraso máximo como 20% do tempo de percurso do trem.

- A classe de trens de minério que abastecem a siderúrgica X , deve ter um tempo final de percurso no máximo 2 horas superior ao tempo de percurso ideal. Neste caso, o atraso máximo é equivalente a 4 horas.

3.5.4 - Formulação Multi-Objetiva "Fuzzy" (FMOF)

- Problema de Max-Min:

Considerando o desenvolvimento das funções de pertinencia [85] que formarão um conjunto de restrições flexiveis, e as restrições estruturais do problema de sequenciamento de trens (para efeito de simplificações, desconsideramos as restrições disjuntivas relativas aos conflitos de ultrapassagem), podemos estabelecer a seguinte formulação multi-objetiva "fuzzy", obedecendo ao princípio de agregação proposto inicialmente por (R.E. Bellman , L.A. Zadeh[1970]):

Max z (PMOF)

$$z \leq B_{k+1} + \frac{B_k - B_{k+1}}{u_{k+1} - u_k} \cdot (u_{k+1} - t_{i,j}) ,$$

para $u_k \leq t_{i,j} \leq u_{k+1}$, $k=1,\ldots,5$ e $i=1,\ldots,n$

$t_{i,j} \geq r_i$, $i=1,\ldots,n$ e j relativo a primeira seção ,

$t_{i,j+1} \geq p_{i,j} + t_{i,j}$, $i=1,\ldots,n$ e j relativo as demais seções,

$t_{i,j} \geq p_{k,j} + t_{k,j}$

ou

$t_{k,j} \geq p_{i,j} + t_{i,j}$

p/ $\forall$ i e k em conflito na seção j,
$j \in R(i)$ e $j \in R(k)$,

$t_{i,j} \geq 0$, p/ $i=1,\ldots,n$ e $j \in R(i)$.

- Problema de Inequações:

Se admitirmos um nível mínimo de satisfação associado aos objetivos do problema (PMOF), podemos considerar a formulação multi-objetiva do mesmo na forma de um conjunto de inequações. Neste caso, introduzimos a formulação anterior a inequação:

$$z \geq B_k \quad , \tag{87}$$

onde B_k se refere ao nível mínimo de satisfação admissível, e o problema se torna determinar uma solução viável que atenda ao conjunto de inequações estabelecidas.

A resolução do problema de sequenciamento de trens na forma de inequações, ou a determinação de uma solução viável que atenda um nível mínimo de satisfação é mais fácil do que a determinação de uma solução ótima. De fato, para valores diferentes de z, considerando X_z como o conjunto de soluções admissíveis do problema, na forma de inequações, podemos garantir a seguinte expressão:

$$z' < z'' < z^* \Longrightarrow X_{z^*} \subset X_{z''} \subset X_{z'} \; . \tag{88}$$

Claramente, se algum valor de z for superior a z^*,o problema é considerado inviável.

- Trabalhos Correlatos:

E.L. Hannan [1981], também, propõe a construção de funções de

pertinência para cada objetivo. Ele, inicialmente, determina para cada objetivo um conjunto de valores associados a graus de pertinência que podem assumir valores no intervalo aberto (0.0..1.0). Em seguida, Hannan constrói uma função de pertinência interpolada, empregando os pares de valores fornecidos e, utilizando uma equação paramétrica, estabelece uma função linear por partes e contínua.

A função objetiva final é formulada segundo (R.E. Bellman e L.A. Zadeh[1970]), ou seja, utilizando o operador "min". Hannan considera, também, a possibilidade de uma combinação linear das funções de pertinência individuais utilizando um esquema de pesos e prioridades pré-estabelecidos.

- Considerações a Respeito da FMOF:

Uma importante característica da formulação multi-objetiva "fuzzy" apresentada, é que a importância relativa dos objetivos dependem da solução considerada, e esta dependência é capturada internamente, pelas funções de pertinência, ao invés da dependência explícita que caracteriza os métodos interativos. Ressalta-se, também, que a importância relativa dos objetivos é um tanto imprecisa, não podendo ser fixada a priori ou pré-estabelecida.

A formulação multi-objetiva do problema, associada as funções de pertinência é uma maneira natural de refletirmos o caráter dinâmico e impreciso do problema de despacho de trens.

A formulação anterior, que considerava a função clássica do problema de "scheduling", não representa o caráter dinâmico do problema de despacho de trens, pois trata as prioridades de trens de uma forma não flexível. A formulação multi-ojetiva permite a definição de uma região flexível, formada pelas funções de pertinência, na qual o tomador de decisão pode exercer um certo tipo de controle.

Segundo C. V. Negoita [1985], a existência de múltiplos critérios ou múltiplos objetivos está associada a falta de decisão gerada por conflitos. A ocorrência de um conflito decisório leva ao surgimento de múltiplas perspectivas. Se houver um único critério ou um único objetivo para orientar o julgamento, certamente, não teremos

conflito decisório e o processo seletivo que determina a escolha de uma solução ótima se torna mais simples.

A formulação "fuzzy" quando aplicada a programação multi-objetiva consiste em um processo "pullback", ou seja, um processo semelhante ao processo de síntese no raciocínio , pois a cada iteração é feita uma avaliação interna dos objetivos segundo os graus de existência (valores das funções de pertinência), permitindo um cálculo subjetivo, uma forma de classificação que é característica do pensamento humano.

O primeiro trabalho que apresenta um aspecto dinâmico ao problema de despacho de trens foi descrito por (Krishna Kant [1981]). Neste trabalho kant desenvolve uma função de avaliação que decide a questão de precedência de trens em um conflito relativo a uma seção em disputa na ferrovia.

A função de avaliação leva em consideração alguns fatores considerados essenciais ao controle de trafego de trens. Entre os principais fatores Kant destacou:

- a prioridade estática dos trens;
- a capacidade dos postos de cruzamento e ultrapassagem;
- a performance dos trens;
- a distância dos trens dos terminais (fim de percurso).

A principal crítica que podemos fazer em relação ao trabalho de Kant é que a função de avaliação é utilizada, somente, para a solução de conflitos localizados, não levando em conta a propagação de decisões presente na elaboração de um plano de horários. Desta forma, a otimalidade das decisões fica seriamente prejudicada. Podemos considerar, também, a existência do caráter subjetivo referente a determinação e desenvolvimento dos fatores.

Segundo H. Simon[1986], uma das principais virtudes dos algoritmos de busca utilizados em solução de problemas está no fato do computador suportar um grande esforço computacional em um tempo de processamento relativamente pequeno. Portanto, no desenvolvimento de um sistema de suporte a decisões, para o controle e gerenciamento de trens, devemos considerar o exame de um número maior de soluções na construção de um plano de rotas.

Talvez, esta seja a principal vantagem que um sistema automático possa ter sobre o controlador que trabalha no despacho de trens. Respondendo, no questionário do capítulo II, a algumas perguntas relativas a viabilidade da implementação deste sistema, verificamos a enorme dificuldade do despachador em examinar um número maior de soluções que envolva alguma forma de raciocínio ao longo do tempo. Raciocinar ao longo do tempo e avaliar o efeito da tomada de decisão significa planejar. A execução de planejamento é, certamente, mais complexa e inteligente que a simples tomada de decisão.

Portanto, é de nosso interesse a realização de um planejamento eficiente, que avalie o efeito da tomada de decisão, e proporcione um plano de horários que orientará o sequenciamento de ações, evitando a ocorrência de conflitos . A tomada de decisões que propicia o controle de trens, é uma tarefa complementar, e será consequência natural da execução deste planejamento.

3.5.5 - Reavaliação de Prioridades

- Reavaliação de Planos:

Segundo E. Charniak e D. Mcdermott [1985], o gerenciamento de planos de ações pode ser decomposto em três problemas fundamentais : geração, seleção e coordenação de planos .

Alguns objetivos básicos, como por exemplo, evitar que o planejamento realizado se torne inviável e minimizar a utilização de recursos devem ser considerados durante o processo de desenvolvimento de um plano. Entretanto, uma das questões mais difíceis que envolve a atividade de planejamento se refere a monitoração e execução de planos, ou seja:

Como monitorar o progresso de um plano ?

Como avaliar a execução de um plano ?

Quando decidir a necessidade de um replanejamento ?

Como garantir o sequenciamento de ações na ocorrência de replanejamento ?

O processo de reavaliação de planos, ou replanejamento, pode ser considerado como um processo "feedback" ou uma realimentação do sistema de planejamento. Um dos aspectos importantes na execução do replanejamento se refere a reavaliação das prioridades dos objetivos por parte do tomador de decisão.

- Percentuais de Reajuste:

Na formulação multi-objetiva "fuzzy" apresentada, podemos realizar a reavaliação de prioridades alterando os intervalos de tempo que servem de suporte para o desenvolvimento das funções de pertinência, sempre que houver a necessidade de se efetuar um replanejamento. Desta forma, admitindo um carater compensatório na reavaliação das prioridades, podemos considerar o emprego das seguintes regras:

Se a função de pertinência do objetivo possui prioridade pouco importante ($1 \geq \mu_{i,j} \geq 0.9$) no momento do replanejamento
Então o atraso máximo calculado será majorado em 100% ;

Se a função de pertinência do objetivo possui prioridade medianamente importante ($0.9 > \mu_{i,j} \geq 0.7875$) no momento do replanejamento
Então o atraso máximo calculado será majorado em 50% ;

Se a função de pertinência do objetivo possui prioridade importante ($0.7875 > \mu_{i,j} \geq 0.6565$) no momento do replanejamento
Então o atraso máximo calculado permaneçe o mesmo;

Se a função de pertinência do objetivo possui prioridade muito importante ($0.6565 > \mu_{i,j} \geq 04522$) no momento do replanejamento
Então o atraso máximo calculado será decrescido em 33% ;

Se a função de pertinência do objetivo possui prioridade máxima ou crítica ($0.4522 > \mu_{i,j} \geq 0$) no momento do replanejamento
Então o atraso máximo calculado será decrescido em 50% ;

Os valores percentuais utilizados na redefinição dos atrasos são números empíricos, e , dependem , certamente, do método interativo proposto ao processo de reavaliação.

Claramente, se a função de pertinência de algum objetivo for

inaceitável, ou seja, ocorrer um atraso superior a 50% do atraso máximo, teremos a necessidade imediata de efetuar um replanejamento.

3.5.6 - Operadores Compensatórios:

H.-J. Zimmermann e P. Zysno [1980] , apresentam uma nova forma de interpretação do "and" linguistico existente em problemas de análise multi-critério.

Assumindo, que a validade do operador "min" proposto por R.E Bellman e L.A. Zadeh [1970] seja admissível como forma de representar o "and" lógico, no sentido de representar a interseção de conjuntos "fuzzy" associados aos objetivos do problema em um ambiente de decisão "fuzzy", teremos a seguinte correspondência, figura 31.

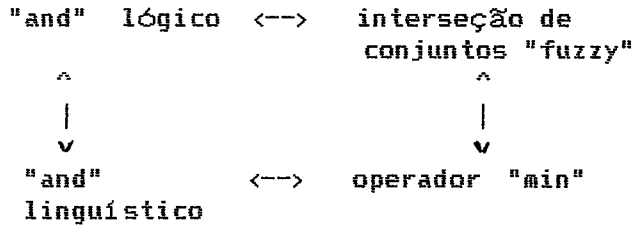


figura 31: correspondência entre operadores.

Entretanto, segundo argumentação de H.-J. Zimmermann [1988] em recente trabalho, baseadas em estudos experimentais realizados por (U. Thole, H.-J. Zimmermann e P. Zysno [1978]) :

. os operadores produto e "mim" não são ideais para representarem o "and" lógico no contexto de tomada de decisões e

. o "and" linguístico nem sempre significa o "and" lógico.

Desta forma, Zimmermann e Zysno[1980] propõe a idéia de um operador compensatório ou um "and" compensatório para expressar o "and" linguístico que representa uma combinação compensatória entre o "and" lógico, que expressa a falta total de compensação, e o "or" inclusivo, que representa a compensação total.

Ou seja, o "and" compensatório é definido como uma combinação geométrica dos operadores "and" lógico (expressando a interseção de conjuntos "fuzzy") e "or" inclusivo (expressando a união de conjuntos

"fuzzy") representados, respectivamente pelo produto e soma algébricos.

Neste caso temos :

$$\mu_{A\Theta B} = \mu_{A\cap B}^{1-y} \cdot \mu_{A\cup B}^{y} \quad , \quad \text{como} \qquad (89)$$

$$A\cup B = \overline{\bar{A}\cap\bar{B}} \quad \text{e} \quad \mu_{A\cap B} = \mu_A \cdot \mu_B \ ,$$

obtemos, para "m" objetivos:

$$\mu_{A\Theta B} = (\Pi_{i=1}^{m} \mu_i)^{1-y} (1 - \Pi_{i=1}^{m}(1-\mu_i))^{y} \quad p/ \quad 0 \leq y \leq 1 \ . (90)$$

O parâmetro "y" é conhecido como grau de compensação . Para y = 0 , teremos a ausência total de compensação equivalente ao "and" lógico e, para y = 1 , teremos a compensação total relativo ao "or" inclusivo.

Segundo Zimmermann [1988], quando pessoas agregam medidas subjetivas em sua mente de modo a avaliarem e tomarem decisões,o tipo de operador utilizado contem um grau de compensação relativo a importância dos critérios de decisão associados a cada medida. Este pensamento, caracterizado pelo conflito decisório, produz um operador real que não pode ser expresso pelo "and" lógico ou pelo "or" inclusivo, pois representa , com certeza, um valor intermediário, fruto do grau de compensação atribuído.

Entretanto, em nossa formulação multi-objetiva "fuzzy" fizemos o uso do "and" logico ,já que o próprio Zimmermann [1978],[1980] argumenta a sua viabilidade no sentido de representar a interseção de conjuntos "fuzzy" em problemas de programação multi-objetiva onde é possível a associação de funções de pertinência e níveis de aspiração aos objetivos do problema .

A utilização do operador "min" para representar o "and" lógico, permitiu elaborarmos a formulação do tipo max-min para o problema de planejamento e "scheduling" , que , efetivamente, propiciou a redução de um problema de multi-objetivo para um problema de objetivo único . Segundo C.V. Negoita [1981], este resultado é de substancial importância para o tomador de decisões, no sentido de que a existência de múltiplos critérios de decisão está associado a falta de

decidibilidade gerada pela ausência , ou acumulo parcial de conhecimento.

Na nossa formulação, permitimos a realização da compensação entre objetivos através do processo de reavaliação de prioridades , onde alteramos, segundo regras empíricas, o intervalo de suporte das funções de pertinência , estabelecendo, a cada processo, um novo conjunto de níveis de aspiração associado aos objetivos do problema. É possível, também, estabelecermos uma forma de compensação entre objetivos de uma mesma classe. Neste caso, utilizamos a média algébrica como forma de estabelecer um tipo de compensação equivalente, já que não ha distinção entre componentes de uma mesma classe.

Caso fizessemos uso do operador compensatório definido por Zimmermann e Zysno [1980], o problema de programação multi-objetiva "fuzzy" seria intratável do ponto de vista matemático, dificultando a elaboração posterior, de heurísticas admissíveis .

3.6 - Técnica de Solução

3.6.1 - Desenvolvimento de um Sistema Físico Simbólico

Nesta seção, apresentaremos o desenvolvimento de um sistema físico simbólico que permitirá a realização de uma busca dirigida no espaço do problema definido segundo a FMOF apresentada anteriormente.

Segundo (M.S. Fox [1989]), um sistema físico simbólico definido por (A. Newell e H.A. Simon [1976]), tem capacidade para realizar uma busca inteligente no processo de solução de problemas, gerando e modificando, progressivamente, estruturas simbólicas até produzirmos uma solução final.

O espaço do problema operacionaliza este conceito de sistema físico simbólico , sendo definido por estruturas representativas (estados), operadores (funções de transformação) e uma função de avaliação (heurística).

- Funções de Transformação:

Vimos, na seção 3.3.3, que podemos considerar o processo de

solução do problema de sequenciamento de trens equivalente a realização de uma busca em um grafo de estados estabelecendo um operador ou uma função de transformação que permite a expansão do mesmo a partir da geração de novos estados nas situações de conflito.

Basicamente, a função de transformação ou operador que atende ao conjunto de restrições de sequenciamento e de precedência do problema , pode ser considerada como um sistema de produção contendo as seguintes regras:

. restrições de sequenciamento:

Se
o instante de ocupação do trem "i" na seção "j" é $t_{i,j}$,
o tempo de percurso do trem "i" na seção "j" é $p_{i,j}$ e
a condição do trem "i" parar no posto adjacente é FALSA
Então
o instante de ocupação do trem "i" na seção "j+1" é igual ao instante de liberação da seção "j" pelo trem "i" mais o tempo necessário para o mesmo percorrer o posto adjacente.

Ou seja:

Se

$t_{i,j}$, $p_{i,j}$ e Condição_Parar_Trem"i" = FALSA
Então

$t_{i,j+1} = t_{i,j} + p_{i,j}$.

Devemos, ainda, para a primeira seção da rota de cada trem , considerar os seguintes fatos:

$t_{i,j} = r_i$, referentes a entrada de cada trem em operação.

. restrições de precedência:

Se
o instante de ocupação do trem "k" na seção "j" é $t_{k,j}$,
o tempo de percurso do trem "k" na seção "j" é $p_{k,j}$ e a
condição do trem "i" parar no posto adjacente é VERDADEIRA
Então
o instante de ocupação do trem "i" na seção "j" é igual ao instante de liberação da seção "j" pelo trem "k" mais o tempo necessário para o trem "i" demarrar do posto adjacente

Ou seja, para trens de sentidos contrários:

Se

$t_{k,j}$, $p_{k,j}$ e Condição_Parar_Trem"i" = VERDADEIRA

Então

$t_{i,j} = t_{k,j} + p_{k,j}$.

ou , para trens de mesmo sentido:

Se

$t_{k,j}$, H_j e Condição_Parar_Trem"i" = VERDADEIRA

Então

$t_{i,j} = t_{k,j} + H_j$.

. determinação de conflitos:

Se
o instante de ocupação do trem "k" na seção "j" está situado entre o instante de ocupação e liberação do trem "i" na mesma seção

Então
a condição do trem "i" e do trem "k" pararem nos postos adjacentes é VERDADEIRA.

Ou seja, para trens de sentidos contrários:

Se

$t_{k,j} + p_{k,j} \geq t_{i,j}$ e $t_{i,j} \geq t_{k,j}$

Então

Condição_Parar_Trem"i" = VERDADEIRA e
Condição_Parar_Trem"k" = VERDADEIRA .

ou, para trens de mesmo sentido:

Se

$t_{i,j} > t_{k,j}$ e $t_{i,j+1} < t_{k,j+1} + H_j$

Então

Condição_Parar_Trem"i" = VERDADEIRA e
Condição_Parar_trem"k" = VERDADEIRA .

Desta forma, podemos considerar o processo de construção da árvore de busca ou geração de estados como um processo de inferência em um sistema de produção consistindo de regras na forma:

Se condição Então ação ,
e um conjunto de fatos que permite a aplicação destas regras.

- Tabela de Tempos:

. conceituação:

Podemos dizer que o método de inferência é de raciocínio progressivo e orientado segundo os fatos existentes, determinando uma operação de composição que utiliza a regra "modus ponens" e atualiza ou modifica a cada ciclo (geração de estado) uma tabela de tempos que representa o instante de ocupação das seções pelos respectivos trens. Segundo Charniak e Mcdermott [1985] podemos considerar esta tabela como uma ferramenta necessária ao gerenciamento de planos de ações no problema de planejamento com raciocínio temporal (envolve a programação de atividades ao longo do tempo).

Desta forma, desenvolvemos uma estrutura, que permite manter atualizado o tempo de ocorrência dos eventos que determinam o estabelecimento de planos de ações para o problema de gerenciamento e controle de trens.

Esta tabela de tempos possui a forma de uma matriz com dimensões nxp, onde "n" se refere ao número de trens e "p" ao número de seções do trecho ferroviário sob planejamento, tendo como componentes os valores das variáveis associadas ao instante de ocupação de cada seção pelos respectivos trens, ou seja:

$$
\begin{array}{lcccc}
 & \text{seção}_1 & \text{seção}_2 & \ldots & \text{seção}_p \\
\text{trem}_1 & t_{1,1} & t_{1,2} & \ldots & t_{1,p} \\
\text{trem}_2 & t_{2,1} & t_{2,2} & \ldots & t_{2,p} \\
\vdots & \vdots & \vdots & & \vdots \\
\text{trem}_n & t_{n,1} & t_{n,2} & \ldots & t_{n,p}
\end{array}
\qquad (91)
$$

. implementação:

Considerando que, na elaboração de um plano de ações, o número de seções e o número de trens sob planejamento são, durante o processo de geração, constantes; desenvolvemos uma implementação estática para a tabela de tempos, baseada no desenvolvimento de um tipo abstrato de dados, tendo como representação um "array" de duas dimensões, com componentes reais.

A tabela foi organizada de modo que os trens de mesmo sentido fossem agrupados conjuntamente, facilitando a identificação de conflitos , conforme os algoritmos de justificativa da seção 3.3.6.

Associado a estrutura da tabela de tempos, foram implementados um conjunto de procedimentos relacionados as operações de: incialização, verificação de conflitos, determinação do próximo conflito, estimativas de atraso máximo e atualizações em função da aplicação das funções de transformação.

No nosso sistema, o conhecimento necessário a solução do problema é um conhecimento procedural na forma de restrições e objetivos e esta representado por um sistema de produção (operadores + tabela de tempo) e um conjunto de funções de pertinência. Portanto, o processo de solução pode ser caracterizado por um método direto que extrai o conhecimento a partir da geração de novos estados e um método subjetivo que avalia situações e controla a geração destes estados. Este processo é identificado como uma exploração (busca) de uma árvore de conhecimento (espaço de estados) e se constitui a base de nosso algoritmo.

- Parametros "Fuzzy":

Se considerarmos os parâmetros $t_{i,j}$ e $p_{i,j}$ não exatos,teremos que estudar uma nova forma de inferência que exprima a imprecisão destes parâmetros.

Seja "x" uma váriavel cujos valores estão representados em um universo discursivo U. Desta forma, x = m , significa que atribuimos a variável "x" um valor "m", sendo "m" um elemento genérico de U de forma

que $m \in U$.

Segundo (L.A. Zadeh[1978]), podemos considerar F uma restrição "fuzzy" a variavel "x" se F atua como um restrição elástica aos valores que possam ser atribuidos a "x". Neste caso, utilizamos a função de pertinência μ_F para interpretar o grau para o qual a restrição representada por F é satisfeita quando atribuimos um valor a variável "x", ou seja:

$$x = m : \mu_F(m) \quad . \tag{92}$$

A existência desta função de pertinência pode ser considerada como uma função de distribuição de possibilidade, ou seja, se F atua como uma restrição "fuzzy", R(x), aos valores de "x", então a proposição x é F, representada por:

$$R(x) = F \ , \tag{93}$$

associa uma distribuição de possibilidade a variável "x", tal que:

$$\Pi_x = R(x) \ . \tag{94}$$

A função de distribuição de possibilidade respectiva é definida como numericamente igual a função de pertinência de F, ou seja:

$$\pi_x \text{ é definido como } \mu_F \ . \tag{95}$$

No nosso sistema de produção, consideramos as funções de pertinência desenvolvidas por Tanaka e Asai[1984] para refletir o caráter impreciso dos parâmetros "fuzzy". Portanto, dado uma regra na forma:

Se
$\tilde{m}$, $\tilde{n}$ e premissa
Então
$\tilde{z} = \tilde{m} + \tilde{n}$,

onde $\tilde{m}$, $\tilde{n}$ e $\tilde{z}$ são parâmetros "fuzzy". Podemos considerar a regra de inferência como uma composição de dois numeros "fuzzy", considerando que o valor da premissa seja verdade. Para tanto, sera necessário

utilizarmos um par de valores {α,c} para representarmos cada parâmetro "fuzzy" associado as variáveis componentes da tabela de tempos.

É interessante observamos que o resultado da adição de dois números "fuzzy" e sempre mais "fuzziness" que os números de origem. Esta consideração é consistente com a idéia de raciocínio aproximado segundo a qual a conclusão é mais nebulosa ou pelo menos tão nebulosa quanto as premissas.

Como consequência natural do encadeamento de regras podemos concluir que a imprecisão dos parâmetros é propagada ao longo do processo de planejamento e representada na tabela de tempos. De fato, quanto maior for o horizonte de planejamento, maior será a imprecisão do plano de ações e, com certeza , maior será a possibilidade de efetuarmos um replanejamento.

A utilização da teoria de raciocínio "fuzzy" decorre do fato de não haver incerteza nas premissas que compõe a condição da regras de produção. O que existe, realmente, é uma certa vagacidade ou imprecisão na determinação de alguns parâmetros decorrente da insuficiência de informações.

3.6.2 - Método de Busca

- Aplicação de Técnicas de Inteligência Artificial:

O método proposto para a solução do problema de sequenciamento de trens, segundo a formulação multi-objetiva "fuzzy", consiste na aplicação de técnicas de Inteligência Artificial (IA). Alguma das aplicações de técnicas de IA se referem a solução de problemas de programação combinatória, especialmente, aqueles problemas considerados NP-completos, ou seja, o esforço computacional cresce exponencialmente com o tamanho do problema. Para a solução destes problemas, utilizamos, geralmente, algoritmos de busca no espaço de estados do problema, até encontrarmos a solução ótima do mesmo ou uma solução razoavel.

A eficiência desta busca , está associada diretamente a capacidade de se utilizar ou não, informações a respeito do conhecimento

do domínio e do objetivo do problema. Portanto, estamos tratanto de algoritmos informados de busca, que, de uma forma ou outra, realizam uma busca dirigida para a solução do problema.

Por exemplo, quanto a elaboração do grafo de estados para a solução do problema de PMOF, propomos a realização de uma busca orientada segundo a resolução de conflitos e a satisfação das restrições flexíveis do problema. Verificamos, também, que as restrições estruturais do problema determinaram a definição dos operadores, bem como, a criação de uma estrutura de representação (estado). Neste sentido, podemos afirmar que o método de solução proposto, constitui um processo de satisfação de restrições, o qual é considerado um processo básico de solução de problemas segundo M.S. Fox [1989] e H. Simon[1983].

Convém lembrarmos que a geração de estados sucessivos , a partir de um estado pai, é realizada através da identificação de um conflito e, consequentemente, da aplicação de um operador específico, que determina a introdução de uma restrição disjuntiva , e, a seguir, a atualização apropriada das tabelas de tempos.

Em geral, os algoritmos de busca em IA encontram boas soluções para problemas de combinatória. Entretanto, nem sempre eles encontram a solução ótima. O desenvolvimento de algoritmos exatos, que garantem a solução ótima , irá depender de certas propriedades da heurística adotada. Como veremos adiante, para uma heurística ser exata, ou seja, encontrar a solução ótima, ela deve ser admissível.

Segundo N.J. Nilsson [1980], a importância dos trabalhos de IA, reside no esforco desenvolvido para atenuar o crescimento exponencial das dificuldades em se resolver um problema, a medida em que aumenta o seu tamanho. Desta forma, a obtenção de bons resultados com um trabalho computacional relativamente pequeno, é mais desejável do que encontrarmos a solução ótima para um trabalho computacional muito maior.

- Métodos Empregados na Solução de Problemas:

Em recente trabalho, H.A. Simon [1983] faz um excelente comentário e estabelece algumas relações sobre as principais técnicas de

solução de problemas em IA, ressaltando a solução de problemas pela utilização de raciocínio, pela utilização de um processo de busca e, também, destaca a solução de problemas de forma a satisfazermos um conjunto de restrições e objetivos, que consiste na técnica mais comumente empregada na solução de problemas em Programação Matemática.

A idéia central na solução de problemas utilizando raciocínio é considerar o processo de raciocínio como a realização de inferências em um sistema formal, caracterizado por uma lógica dedutiva baseada no cálculo de predicados. Segundo Simon, o principal aspecto negativo desta técnica se deve ao caráter não monotônico do conhecimento formalizado e o caráter exaustivo do processo de inferência, que leva, certamente, a uma explosão combinatória.

Segundo Simon, na solução de alguns problemas , especialmente problemas com formulação matemática , o processo de raciocínio difere um pouco da lógica dedutiva, ou seja, além de se utilizar de alguns axiomas e de algumas regras de inferência do cálculo de predicados , o processo de raciocínio utiliza-se de procedimentos de inferência mais poderosos que podem ser mais convenientemente representados por um conjunto de operadores ou funções de transformação de estados.

Desta forma, Simon considera a lógica dedutiva como um sub-conjunto do processo de raciocínio, que por sua vez é considerado como uma forma de processo de busca, onde um conjunto de regras de inferência funciona como operadores. Neste caso, a fim de reduzirmos a complexidade da busca, devemos estabelecer uma forma de controlar a aplicação das regras de inferência (operadores), dai a utilização cada vez mais crescente de estratégias de controle, conhecimento procedural, macro-operadores, meta-regras, etc., em bases de conhecimento.

Quando resolvemos um problema utilizando um processo de busca, consideramos a representação do espaço de estados, e o processo de inferência consiste em provocarmos mudança de estados pela aplicação sucessiva de operadores. A estratégia basica, que controla e dirige o processo de busca, é bem simples, e se baseia na idéia de algum método fraco de busca em I.A. . Podemos citar como exemplo, a idéia do método de eliminação de diferenças (entre o estado incial e o estado solução) de forma sucessiva e monotônica, até atingirmos o estado que representa

a solução do problema. O nosso sistema fará uso dos metodos A*, IDA* e também, de um método que realiza sucessivas buscas em profundidade.Estes métodos se baseiam nas estratégias de ramificação apresentadas na seção 3.3.3 .

3.6.3 - Função de Avaliação (FA)

- Algoritmo BF*:

Se conseguirmos associar os nossos objetivos, a uma função de avaliação única, podemos ordenar os pontos examinados em uma reta, e escolhermos o de maior valor numérico, tratando de um problema de maximização.

Uma solução natural seria examinarmos o menor número possível de alternativas, através de um algoritmo de busca direcionado no espaço de estados e utilizarmos a função de avaliação como critério de ordenamento.

Os algoritmos de busca informada, que se comportam segundo a primeira estratégia de geração de estados apresentada na seção 3.3.3, são conhecidos como BF*. Esta classe de algoritmos foi formalizada por R.Dechter e J.Pearl [1985]), no sentido de que o melhor estado é aquele que possui o melhor valor de "custo", daí o nome "best first".

A formulação multi-objetiva "fuzzy" apresentada anteriormente, permitiu tratarmos um problema de programação multi-objetiva como um problema de programação com um objetivo único.

Portanto, se garantirmos que a função de avaliação (valor de "z") apresente valores monotonicamente decrescentes a medida que tornamos mais restrito o problema (inserimos restrições de precedência) , podemos determinar limites inferiores que serão úteis na orientação do processo de busca.

Desta forma, se conseguirmos estabelecer um limite para o valor de "z" associado a uma solução viável, podemos afastar do processo de busca todo estado que apresentar um valor para a função de avaliação inferior ao limite estabelecido.

Vale lembrarmos, que este critério de poda é semelhante ao critério utilizado no algoritmo "branch and bound"; entretanto, no problema de formulação multi-objetiva, que considera a utilização do operador "min", o mesmo se torna mais eficiente,já que o mesmo se aplica a todos os objetivos do problema.

- Componente Heurística:

Se considerarmos o valor da variável $t_{i,j}$ como o tempo associado ao instante da ocorrência dos conflitos, relativos a introdução das restrições de precedência, não podemos garantir o caráter monotônico decrescente da função de avaliação. Portanto, será necessário a consideração de um tempo extra, um segundo componente, relativo a parte heurística desta mesma função.

Desta forma, o primeiro componente representa uma medida do custo do estado inicial até um estado "current" (associado a um conflito) . Enquanto que, o segundo componente, representa uma estimativa do custo adicional do estado "current" até o estado solução. É na determinação desta componente heurística que exploramos o conhecimento sobre o domínio do problema.

Segundo E. Rich [1983], a adoção de uma heurística é uma técnica que visa melhorar a eficiência do processo de busca. Entretanto, se a mesma nao for admissível, afetamos a otimalidade da solução.

Desta forma, para um estado qualquer , representado por X_n , podemos determinar o valor da função de avaliação segundo a expressão:

$$FA(X_n) = g(X_n) + h(X_n) . \qquad (96)$$

onde: $g(X_n)$ se refere ao custo atual e $h(X_n)$ é uma estimativa do custo adicional até o estado solução.

- Caráter Monotônico Decrescente:

No nosso problema, é fácil verificarmos que a parte heurística da função é relativa a previsão mais otimista do percurso final dos trens; e, consequentemente, relativa a previsão mais pessimista da

função de avaliação.

Portanto,se considerarmos o valor da variável $t_{i,j}$ na avaliação das funções de pertinência como referente a última seção da respectiva rota R(i) (última coluna da tabela de tempos), estamos utilizando a estimativa de custo de maior valor.

Neste caso, estamos garantindo, também, o caráter monotônico decrecente da função de avaliação, como veremos a seguir, se a parte heurística da função for considerada como a realização do percurso restante dos trens em condições ideais (sem ocorrência de atrasos) .

Prova:

Considerando o valor da função de avaliação segundo a formulação multi-objetiva "fuzzy", temos:

$$FA = \text{Max } z \qquad (97)$$

$$z \leq \mu_i(t_{i,j}) \text{ , para } i=1,\ldots,m \quad .$$

Como os tempos $t_{i,j}$ se referem ao instante de ocupação da última seção de cada rota respectiva, e, considerando que as funções de pertinência μ_i (.) são monotonicamente decrescentes para valores crescentes de $t_{i,j}$ (lembrando que estes valores são sempre subestimados) , garantimos o caráter monotônico decrescente da função de avaliação ■

- Consistência e Admissibilidade:

P.E. Hart, N.J. Nilsson e B. Raphael [1968] desenvolveram uma base formal para a determinação de heurísticas admissíveis para problemas de busca em um grafo de estados, no qual se deseja determinar o melhor caminho que leva a uma solução ótima. Determinar o melhor caminho, significa gerar o menor numero possível de estados necessários para atingirmos a solução ótima, aproveitando-se de informações do domínio do problema.

O algoritmo que utiliza a função de avaliação na forma [96] é conhecido como A*. O algoritmo A* difere do BF*, basicamente, quanto ao cômputo da FA.

. admissibilidade:

O algoritmo A* é dito admissível, quando encontra a solução ótima do problema. Segundo (Hart, Nilsson e Raphael [1968]) o algoritmo A* é admissível, em um problema de maximização, se:

$$h(X_n) \geq h^*(X_n) \text{ para todo estado } X_n, \qquad (98)$$

onde: $h^*(X_n)$ se refere ao custo do caminho ótimo do estado X_n até a solução ótima. Significando que a estimativa de custo até a solução ótima deve ser pessimista (acima do custo real).

Prova:

Considerando a expressão [97] da função de avaliação, temos, para um estado X_n:

$$FA(X_n) = Min_i \ \mu_i(t_{i,j}) = \mu_k(t_{k,n}) \ , \qquad (99)$$

onde: "n" se refere a última seção da rota R(k).

Dado ao aspecto monotônico decrescente, da função de pertinência , e a expressão [96], temos, para um estado X_n:

$$FA(X_n) = g(X_n) + h(X_n) = \mu_k(t_{k,j} + (t_{k,n} - t_{k,j})), \qquad (100)$$

onde: "k" se refere a seção associada ao conflito do estado X_n.

Portanto, de [100], temos que:

$$h(X_n) = \mu_k(t_{k,j} + (t_{k,n} - t_{k,j})) - \mu_k(t_{k,j}) = \qquad (101)$$

$$= \mu_k(t_{k,n}) - \mu_k(t_{k,j}) \ .$$

Como $t_{k,n} = t_{k,j} + t$, garantindo que "t" seja a estimativa mais otimista de tempo, garantimos que $h(X_n)$ é a estimativa heurística de maior valor, satisfazendo, portanto, a expressão [98] ■

. consistência:

O algoritmo A* é dito consistente, em um problema de maximização, se, para cada par de estados X_m e X_n (onde X_m é sucessor de X_n), a seguinte relação é satisfeita:

$$h(X_n) - h(X_m) \geq g(X_m) - g(X_n) \qquad (102)$$

Prova:

Considerando o caráter monotônico decrescente de nossa função de avaliação, temos, para os dois estados, X_n e X_m:

$$FA(X_m) \leq FA(X_n), \qquad (103)$$

Portanto, segundo a expressão [96]:

$g(X_m) + h(X_m) \leq g(X_n) + h(X_n)$, satisfazendo a expressão [102] ■

Desta forma, se o algorítmo expande sempre, o estado da lista de abertos de maior valor (problema de maximização), o mesmo terminará em um tempo finito, com a solução ótima.

Hart, Nilsson e Raphael mostraram, que, se o A* é consistente, então cada estado X_n selecionado para fechar (gerar estados sucessores) tem valor de $FA \geq C^*$, onde C^* é igual ao custo do caminho ótimo, e todos os estados com valor de $FA > C^*$ serão fechados. Significando que, quanto melhor for a heurística (menor o valor de $h(X_n)$), menor será o número de estados fechados, sendo que, para $h(X_n) = h^*(X_n)$, só serão fechados os estados do caminho ótimo. Neste caso a heurística é denominada perfeita.

3.6.4 - Viabilidade das Tabelas de Tempos

- Consistência do Conjunto Ativo e da Base de Conhecimento:

Na seção 3.3.5 analisamos a questão da restauração da viabilidade do sistema de restrições, quando da introdução de uma nova restrição de precedência. Consideramos, também, a possibilidade de realizarmos uma troca de restrições, com a retirada da restrição de sequenciamento do sistema, permitindo a representação das variáveis do problema em uma tabela de tempos. Do ponto de vista matemático, este procedimento será viavél se garantirmos que a variável de folga (associada a restrição de sequenciamento) não retorna ao conjunto ativo do problema (a restrição não volta a ser ativa).

Considerando o sistema de produção apresentado no seção 3.6.1

, a condição básica a ser verificada, que garante a viabilidade do procedimento é que, uma vez uma determinada regra tenha sido aplicada ,não podera ocorrer, posteriormente, a mudança de uma premissa que invalide a aplicação anterior desta regra, ou seja, o sistema deve preservar a verdade dos fatos e as inferências realizadas. .

Para exemplificarmos melhor as considerações feitas acima, vamos imaginar a seguinte situação:

Suponha que num instante "ti", qualquer, haja a previsão de conflito entre dois trens "i" e "k" em um trecho "j" da ferrovia. Neste caso, considerando que a condição de parada do trem "i" seja VERDADEIRA, teremos:

$$t_{i,j} = t_{k,j} + p_{k,j} \qquad (104)$$

Entretanto, se considerarmos, posteriormente, que a solução de um conflito em um instante "t2", envolvendo o trem "i" , de tal forma que t2 < ti (este segundo conflito é cronologicamente anterior ao primeiro) tenha um caráter multi-satisfatório em relação ao primeiro conflito, ou seja, quando da resolução do segundo conflito provocamos a não ocorrência do primeiro, teremos a condição de parada do trem "i" no posto adjacente ao trecho "j" FALSA. Neste caso, o instante de ocupação do trem "i" na seção "j" será recalculado na forma:

$$t_{i,j} = t_{i,j-1} + p_{i,j-1} \text{ ,} \qquad (105)$$

contrariando, portanto, o cálculo realizado anteriormente.

- Propriedades do Planejamento Clássico:

Para garantirmos a viabilidade da tabela de tempos ou o sistema de restrições do problema na forma reduzida, faremos alguns comentários a respeito da técnica de síntese de planos desenvolvida por (E.P.D. Pednault [1986]) que foi abordada no trabalho de (H. A. Kautz e E.P.D. Pednaut [1988]) sobre planejamento e reconhecimento de planos.

A técnica de síntese é baseada em duas propriedades clássicas do problema de planejamento. A primeira afirma que o estado de um sistema só é modificado quando executamos uma determinada ação. A segunda

propriedade afirma que um plano deve ser finito. Consequentemente , se alguma condição é verdadeira em algum ponto durante a execução de um plano e falsa, em um instante posterior, então,alguma ação que alterou o estado da condição, foi executada por último neste intervalo de tempo.

Estas propriedades permitem a definição do seguinte teorema (H. A. Kautz e E.P.D. Pednault [1988]):

Uma condição "p" é verdadeira em algum instante "t" durante a execução de um plano, se e somente se, uma das afirmativas forem verdadeiras:

1 - Uma ação é executada anteriormente ao instante "t" e provoca a condição "p" ser verdadeira e "p" permanece verdadeira até o instante "t";

2 - A condição "p" é verdadeira no estado inicial e permanece verdadeira até o instante "t".

- Soluções de Conflitos na Ordem Cronológica:

Desta forma, se quisermos garantir em nosso sistema de produção que uma premissa se mantenha verdadeira em um instante "t", não podemos admitir que determinada ação, que provoque a mudança da premissa, aconteça. Uma solução, bastante simples, para o problema acima é garantirmos a execução de ações, ou a resolução de conflitos, na ordem natural (cronológica) de ocorrência.

O teorema apresentado anteriormente é utilizado como base para uma técnica de planejamento incremental na qual as ações são inseridas em um plano de forma sucessiva, sempre que for necessário alcançarmos certos objetivos.

No problema de sequenciamento de trens, podemos utilizar esta técnica para a geração de planos parciais (planos viáveis até a execução da última ação ou resolução do último conflito). Conforme abordado anteriormente, na fase introdutória do trabalho, a capacidade de elaborarmos planos parciais é de fundamental importância para o bom funcionamento de um sistema de suporte a decisão para o gerenciamento e controle inteligente de trens.

3.6.5 - Tratamento de Conflitos

- Hierarquia de conflitos:

S.E. Cross [1985] apresenta um trabalho no qual algumas considerações são feitas a respeito do controle de tráfego aéreo, e que são de interesse ao gerenciamento e controle de tráfego ferroviário. Segundo Cross, a experiência do controlador em resolver possíveis colisões (conflitos) pode ser medida pela capacidade de :

1- Reconhecer situações elegantes (planos) para a resolução de conflitos. O reconhecimento de um plano envolve: interpretar o cenário de tráfego, reconhecer conflitos (prevenir colisões) e elaborar um plano de ações;

2 - Relaxar restrições e

3 - Elaborar um plano de ações que satisfaça múltiplos objetivos e restrições (plano multi-satisfatório).

O objetivo mais importante do controle de tráfego aéreo é prevenir a ocorrência de colisões. Uma forma de satisfazer múltiplos objetivos é conseguir um planejamento de decisões que resolva , simultaneamente, um número maior de conflitos. Por exemplo, um mesmo avião pode estar em conflito com duas outras aeronaves que tenham planos de vôo similares. Neste caso, o reconhecimento de uma interação entre dois objetivos pode permitir ao sistema de controle a possibilidade de , com somente uma decisão (desviar a rota do primeiro avião), satisfazer dois objetivos, evitando a ocorrência de duas colisões.

No controle de tráfego ferroviário a capacidade do controlador , bem como, a estratégia de elaboração de planos são muito semelhantes ao que foi descrito anteriormente. A situação de interação de objetivos ocorre sempre que um trem está em conflito com um grupo de trens (onda) que se movimentam em sentido contrário. Neste caso, poderemos, pela supressão de conflitos, reduzir a dimensionalidade do espaço de estados durante o processo de busca. Entretanto, os objetivos de um sistema desenvolvido para dar suporte ao gerenciamento e controle do tráfego de trens, são mais complexos do que a simples prevenção de colisões (bloqueios).

A estrutura que representa a situação, ou cenário, do problema de controle de tráfego aéreo é uma rede semântica onde os nós representam as aeronaves e os arcos representam diferentes tipos de conflitos. É fornecida, também, uma tabela com dados relativos aos próximos conflitos na ordem cronológica.

Cross propõe em seu trabalho, um método de solução de conflitos segundo a estratégia "top-down", ou seja, procurando satisfazer, simultaneamente, múltiplos objetivos. Desta forma, ao invés do estilo tradicional de solução de conflitos par a par, o método proposto tenta identificar a ocorrência de meta-aeronaves como uma forma de reduzir a complexidade da estrutura de conflitos.

Entretanto, para o sistema ferroviário, conforme já ressaltamos, esta estratégia somente será útil na redução do espaço de estados do problema. Respondendo as questões 3 e 4 do questionário, verificamos que o controlador trabalha segundo o estilo tradicional, par a par, procurando resolver os conflitos segundo a ordem cronológica (os conflitos mais proximos são considerados os mais prioritários).

- Estimativa ou Projeção do Instante:

Para determinarmos uma hierarquia de conflitos, ou seja, para sabermos qual o próximo conflito que deverá ocorrer, devemos realizar uma pesquisa na tabela de tempos. Esta pesquisa deve, inicialmente, identificar todos os conflitos e selecionar aquele que apresentar o menor tempo de ocorrência . Isto implica, necessariamente, em uma pesquisa em tempo polinomial, segundo a análise feita na seção 3.3.6 .

Para identificarmos o tempo de ocorrência de um conflito , desenvolvemos uma expressão analítica que calcula a provável interseção das retas que representam a curva de velocidade dos trens , ou seja :

Suponha que o trem "i" esteja em conflito de cruzamento com o trem "k" em um trecho "j" da ferrovia, o tempo de ocorrência do conflito, ilustrado pela figura 32, será dado pela expressão:

$$t = 1.0 / (((t_{i,j+1} - t_{k,j})/((t_{k,j-1} - t_{i,j}).(t_{i,j+1} - t_{i,j}))) + (1.0/(t_{i,j+1} - t_{i,j}))) + t_{i,j} \quad (106)$$

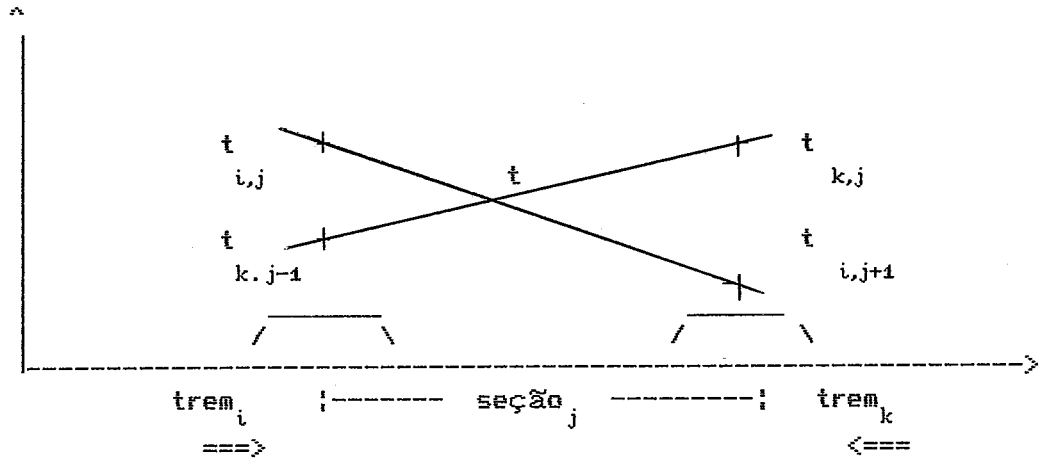


figura 32: situação de conflito (cruzamento).

claramente, para:

$t_{i,j+1} = t_{k,j} \Rightarrow t = t_{i,j+1}$, e

$t_{k,j-1} = t_{i,j} \Rightarrow t = t_{i,j}$.

Agora, suponha que o trem "i" esteja em conflito de ultrapassagem ou manutenção do "headway" com o trem "k" em um trecho "j" da ferrovia, o tempo de ocorrência do conflito, ilustrado pela figura 33 , será dado pela expressao:

$$t = 1.0 / (((t_{i,j+1} - t_{k,j+1})/((t_{k,j} - t_{i,j}).(t_{i,j+1} - t_{i,j}))) + (1.0/(t_{i,j+1} - t_{i,j}))) + t_{i,j} \qquad (107)$$

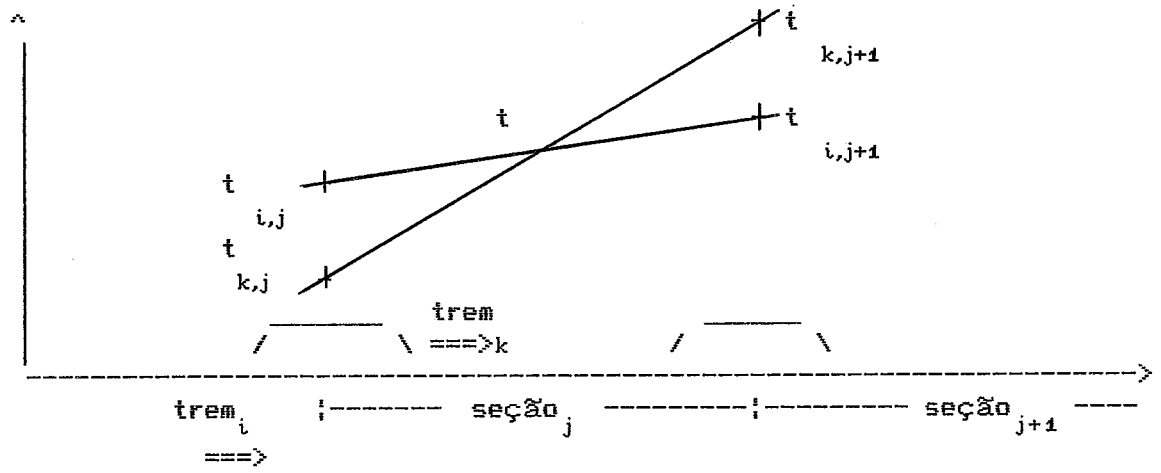


figura 33: situação de conflito (ultrapassagem).

claramente, para:

$t_{i,j+1} = t_{k,j+1} \Rightarrow t = t_{i,j+1}$, e

$t_{k,j} = t_{i,j} \Rightarrow t = t_{i,j}$.

3.6.6 - Algoritmos Propostos

- Algoritmo A*:

Em primeiro lugar , vamos apresentar um algoritmo de busca compatível com a primeira estratégia de ramificação apresentada na seção 3.3.3. Este algoritmo, pelo fato de adotar uma estimativa otimista do custo final na função de avaliação , determinando uma heurística admissível, pode utilizar um limite inferior para eliminar estados indesejáveis. O algoritmo é aplicado a formulação Max-Min e termina com uma solução ótima e viável (sem conflitos). Em pseudo-código, temos:

```
Início
  Inicia lista de ABERTOS;
  Gera Raiz ( tabela de tempos inicial );
  Determina atrasos para cada objetivo;
  Constrói as funções de pertinência para cada objetivo;
  Calcula limite inferior para a FA;
  Determina critérios de poda;
  Identifica  próximo conflito ( raiz );
  Se não ha conflito  Então
    fim(estado):= TRUE
  Senão
    fim(estado):= FALSE
  Fim Se;
  Enquanto (não fim(estado)) Faça
    Aplique operador sobre estado gerando 2 estados
    sucessores;
    Para cada estado gerado Faça
      Atualize a tabela de tempos;
      Calcule o valor da FA;
      Verifique critérios de poda e limite inferior;
      Se  (não há poda) e (eliminação) Então
        Insira na lista de ABERTOS na ordem decrescente de FA
      Senão
        Elimine  ( estado gerado )
      Fim Se
    Fim Para;
    Elimine ( estado pai );
    Selecione o primeiro estado ( pai ) da lista de ABERTOS;
    Identifique próximo conflito ( estado pai );
    Se não há conflito  Então
      fim(estado):= TRUE
```

```
    Senão
      fim(estado):= FALSE
    Fim Se;
  Fim Enquanto
Fim.
```

- Algoritmo com Sucessivas Buscas em Profundidade:

Neste caso, apresentamos um algoritmo compatível com a terceira estratégia de ramificação apresentada na seção 3.3.3. Este algoritmo realiza sucessivas buscas em profundidade, melhorando, a cada iteração o valor do limite inferior. Ao invés de fazermos o "bactracking" cronológico, retornamos ao melhor estado da lista de abertos. O algoritmo é aplicado a formulação de inequações e termina com uma solução satisfatoria (se existir) e viável (sem conflitos). Em pseudo-código temos:

```
Início
  Inicia lista de ABERTOS;
  Gera Raiz ( tabela de tempos inicial );
  Determina atrasos para cada objetivo;
  Construa as funções de pertinência para cada objetivo;
  Forneça valor aceitável para FA;
  Determina critérios de poda;
  Identifica  próximo conflito ( raiz );
  Se não há conflito Então
    fim(estado):= TRUE
  Senão
    fim(estado):= FALSE
  Fim Se;
  Enquanto (não fim(estado)) Faça
    Aplique operador sobre estado gerando 2 estados
    sucessores;
    Para cada estado gerado Faça
      Atualize a tabela de tempos;
      Calcule o valor da FA;
      Verifique critérios de poda e limite inferior;
      Se  (não há poda) e (eliminação) Então
        Insira na lista de ABERTOS na ordem decrescente de FA
      Senão
        Elimine  ( estado gerado )
      Fim Se
    Fim Para;
    Calcule Limite Inferior ( estado pai ), produzindo um
    estado final;
    Se Limite Inferior ≥ Valor Aceitável Então
      fim(estado):= TRUE
    Senão
      fim(estado):= FALSE
    Fim Se;
    Elimine  ( estado pai );
    Selecione o primeiro estado ( pai ) da lista de ABERTOS;
```

```
      Identifique próximo conflito ( estado pai );
    Fim Enquanto
  Fim.
```

- Cálculo do Limite Inferior:

Para o cálculo do limite inferior realizamos uma busca em profundidade, direcionada pelo uso de heurísticas na escolha do operador, até produzirmos um estado final sem conflitos. Em pseudo-código temos:

```
Limite Inferior ( estado );
Início
  Identifique próximo conflito (estado );
  Se não há conflito
    fim(estado):= TRUE
  Senão
    fim(estado):= FALSE
 Fim Se;
 Enquanto (não fim(estado)) Faça
   Aplique operador sobre estado gerando 2 estados
   sucessores;
   Para cada estado gerado Faça
     Atualize a tabela de tempos;
     Calcule o valor da FA;
     Verifique critérios de poda e limite inferior;
     Se  (não há poda) e (eliminação) Então
       Determine preferência, gerando um estado sucessor
     Senão
       Retorne
     Fim Se
   Fim Para;
   Identifique próximo conflito ( estado sucessor );
   Se não há conflito  Então
     fim(estado):= TRUE
   Senão
     fim(estado):= FALSE
   Fim Se
 Fim Enquanto
Fim.
```

Para a determinação de preferência entre dois operadores , utilizamos dois critérios:

. melhor valor da função de avaliação (FA), e

. escolha do operador que provoque menor atraso de tempo na solução do conflito , considerando as duas possibilidades de decisão.

- Aspectos Computacionais:

. complexidade em tempo:

Na implementação dos algorítmos de busca, consideramos, para a representação e manipulação dos estados, o desenvolvimento de um tipo abstrato com estrutura dinâmica, contendo as informações sobre os valores da função de avaliação, sobre a tabela de tempos associada ao estado, posicionamento na árvore de busca (nível), informações sobre antecessores do estado (solução dos conflitos anteriores) e informações adicionais, que serão tratadas posteriormente. Na sintaxe de Modula-2, temos:

```
TYPE PtState = POINTER TO TipState;
     TipState = RECORD
                  high, low: CARDINAL; (* endereço da      *)
                  ptr: ADDRESS;        (* tabela de tempos *)
                  nivel: CARDINAL;
                  FuncPert: TipArray;
                  ValFun: REAL;
                  conflito: ARRAY [0..MaxConf] OF RECORD
                    trem1,trem2,NumSEção,CodAção: CARDINAL;
                    instante: REAL
                                                  END;
                  aguarda: ARRAY [0..MaxConf] OF RECORD
                    trem, seção: CARDINAL
                                                 END;
                  bloqueio: ARRAY [1..2] OF RECORD
                    trem, cod, PosSeção: CARDINAL;
                    janela: RECORD
                      cheg, sai: REAL
                           END
                                        END
                END;
```

Consideramos, também, a implementação de procedimentos associados as funções de : geração de raiz, geração de estados, critérios de poda, operadores, etc. Este módulo, trabalha conjuntamente, com um módulo que implementa a tabela de tempos, e com um módulo funcional que implementa os procedimentos necessários ao cálculo da função de avaliação, definição de objetivos e construção das funções de pertinência.

Para representarmos a lista de estados abertos, implementamos um tipo abstrato lista, contendo uma estrutura dinâmica duplamente encadeada, que armazena enderêços relativos as variáveis do tipo estado, tendo, portanto, concepção genérica.

Esta lista é ordenada segundo o valor de uma chave que corresponde, na lista de abertos, ao valor da função de avaliação dos

respectivos estados. Na sintaxe de Modula-2, temos:

```
TYPE link = POINTER TO linkage;
     linkage = RECORD
                 succ, pred: link;
                 pinfo: ADDRESS;   (* PtState *)
                 key: REAL
               END;

     lista = RECORD
               head: link;
               tam, TamMax: CARDINAL;
               vazia, cheia: BOOLEAN
             END;
```

A lista de estados abertos funciona como uma árvore hierárquica, da qual retiramos, sempre, o estado de maior valor, que se encontra na primeira posição. Desta forma, a cada operação de inserção, necessitamos de realizar uma pesquisa sequencial, com complexidade O(n), superior, portanto, a complexidade O(logn) de uma pesquisa binária, ou de um procedimento de inserção em árvore hierárquica (R. Terada [1982]).

Porém, dado o aspecto da função de avaliação, consideramos a implementação da lista de abertos no estilo de listas indexadas (J.G. Vaucher e P. Durval [1975]) reduzindo a complexidade de inserção para O ($\sqrt{N}$).

Esta implementação consiste em dividirmos a lista de estados abertos em "n" sub-listas, em função do número máximo de estados, ao qual chamamos de N.

Prova:

Considerando, que a função de avaliação assume valores no intervalo (lim..1.0), onde lim se refere ao limite inferior de poda, podemos distribuir os estados ao longo das "n" sub-listas, considerando os seguintes procedimentos em pseudo-código:

```
Procedimento RetornaEstado();
Início
  Selecione estado ( lista[inf],êxito);
  Se (não êxito) e (inf < n) Então
    inf:= inf + 1;
    RetornaEstado
  SenãoSe (não êxito) Então
     fim(estado):= TRUE
  Senão
     fim(estado):= FALSE
  Fim
Fim.
```

```
Procedimento SelecionaÍndice( ValFun: REAL ): CARDINAL;
Início
  Para i <- inf até n Faça
    Se ValFun > (n - i.(1.0 - lim))/n Então
      Retorne i
    Fim
  Fim
Fim.
```

Desta forma, se considerarmos a função de complexidade média, tendo como argumento o número máximo de estados, e uma hipótese simplificadora que admite uma distribuição uniforme dos estados, teremos para o procedimento de inserção:

$$T(N) = N / 2.n + n / 2, \qquad (108)$$

como, para $d\,T(N)/dn = 0$, $n = \sqrt{N}$, concluímos:

$$T(N) = \sqrt{N} \qquad (109)$$

Entretanto, se observarmos que as sub-listas iniciais são fechadas ($inf \geq 1$) durante a execução do algoritmo, teremos:

$$T(N) \leq \sqrt{N}, \qquad (110)$$

e, portanto, de ordem máxima $O(\sqrt{N})$ ■

Na verdade, podemos determinar a sub-lista apropriada, em tempo constante, através da função:

```
indice := TRUNC (n.(1.0 - ValFun)/(1.0 - lim)) + 1        (111)
```

. complexidade em espaço:

Para determinamos o número máximo de estado possíveis na lista de abertos, consideramos a memória utilizada pelo programa e a memória requerida para cada estado.

A memória utilizada pelo programa, consiste, basicamente:

. 300 kbytes de código e memória global;
. 8 a 64 kbytes de memória de pilha;
. 30 kbytes de memória de regras (opcional);
. 38 kbytes de memória do sistema operacional;

. 64 kbytes de memória de processos (simulação do planejamento).

Considerando, ntc o número total de classes, ntt o número total de trens e nts o número total de seções, temos a seguinte memória requerida pelas "frames":

$$ntc(21 + 8.nts) + 20.ntt + 50.nts \quad \text{kbytes}, \tag{112}$$

fazendo $ntc=10$, $ntt=20$ e $nts=25$, temos:

. 3.8 kbytes de memória de "frames".

Sendo a memória RAM convencional, utilizada pelo DOS, de 640 kbytes, e, utilizando uma memória de pilha de 12 kbytes, restam, portanto, de 190 a 220 kbytes de memória livre para o armazenamento da lista de estados.

A memória requerida por um estado é igual a:

$$8.(ntt(1+nts+2.ntt)+ 9.25) \quad \text{kbytes}, \tag{113}$$

portanto,utilizando os valores anteriores,temos,aproximadamente, 10 kbytes por estado, e um maximo de 22 estados na lista de abertos.

Para uma aplicação, envolvendo 12 trens em 20 seções, teríamos, aproximadamente, 4.4 kbytes por estado,e um máximo de 50 estados na lista de abertos.

Lógicamente, que o número máximo de estados é imcompatível com a utilização do algoritmo A* (um número razoável seria em torno de 100 estados). Portanto, considerando que a tabela de tempos representa quase 50% da área de memória requerida (para o primeiro caso 4 kbytes) e, considerando um limite máximo de 80 conflitos a serem resolvidos, teríamos uma redução da memória de cada estado para 1.5 kbyte.

Neste caso, seria possível armazenarmos até 150 estados na lista de abertos. Para tanto, guardaríamos as tabelas de tempos em um arquivo com registros de tamanho fixo na memória virtual (expandida até 1.4 Mbyte), as quais seriam representadas nos estados, pelos respectivos endereços relativos (RRN). Como, cada tabela de tempos ocupa uma área de 4 kbytes, poderíamos armazenar até 350 estados na memória virtual.

- Exemplo Aplicativo (A*):

Deteminar o plano de horários ótimo do seguinte problema de sequenciamento de trens, caracterizado pela tabela de tempos inicial:

1 - Tabela de tempos inicial (raiz):

	seção$_1$	seção$_2$	seção$_3$	seção$_4$	$\mu_i(t_{i,j})$
trem$_1$	0.0	0.75	1.50	2.25	1.0
trem$_2$	3.0	2.0	1.0	-	1.0
trem$_3$	4.0	3.0	2.0	1.0	1.0

Obs: A tabela de tempos inicial é construída a partir dos tempos de ocupação da primeira seção e dos tempos de percurso, com a aplicação do operador relativo as restrições de sequenciamento. Neste exemplo não consideramos a imprecisão dos parâmetros.

2 - Definição dos objetivos e construção das funções de pertinência:

trem 1 (em função do atraso máximo):
atraso máximo = 2 horas (conflito com trens 2 e 3);

trem 2 (em função do atraso máximo):
atraso máximo = 0.75 horas (conflito com o trem 1);

trem 3 (critério):

atraso máximo = 40% tempo do percurso ideal (c_3) = 2.40 horas.

. funções de pertinência:

nível	μ_k trem$_1$	μ_k trem$_2$	μ_k trem$_3$	B_k
1	2.25	3.0	4.0	1.0
2	2.45	3.075	4.24	0.9
3	2.65	3.15	4.48	0.7875
4	2.85	3.225	4.72	0.6565
5	3.05	3.30	4.96	0.4522
6	3.25	3.375	5.20	0.0

3 - Critérios de poda:

. Não há limite inferior;

. Todo estado com valor de FA abaixo de 0.4522 será eliminado.

4 - Geração de estados:

estado inicial (raiz) , FA = 1.0

próximo conflito:
trem 1 e trem 2 na seção 3 (1.714 horas)

primeiro estado (trem 1 aguarda):

tabela de tempos atualizada:

	seção$_1$	seção$_2$	seção$_3$	seção$_4$	$\mu_i(t_{i,j})$
trem$_1$	0.0	0.75	2.0	2.75	0.722
trem$_2$	3.0	2.0	1.0	-	1.0
trem$_3$	4.0	3.0	2.0	1.0	1.0

FA = 0.722

segundo estado (trem 2 aguarda):

tabela de tempos atualizada:

	seção$_1$	seção$_2$	seção$_3$	seção$_4$	$\mu_i(t_{i,j})$
trem$_1$	0.0	0.75	1.5	2.25	1.0
trem$_2$	3.5	2.5	1.5	-	0.2261
trem	4.0	3.0	2.0	1.0	1.0

FA = 0.2261 (inaceitável) , estado eliminado

elimina raiz

seleção do primeiro estado, FA = 0.722

próximo conflito:
trem 1 e trem 3 na seção 3 (2.4285 horas)

terceiro estado (trem 1 aguarda):

tabela de tempos atualizada:

	seção$_1$	seção$_2$	seção$_3$	seção$_4$	$\mu_i(t_{i,j})$
trem$_1$	0.0	0.75	3.0	3.75	0.2265
trem$_2$	3.0	2.0	1.0	-	1.0
trem$_3$	4.0	3.0	2.0	1.0	1.0

FA = 0.2265 (inaceitável), estado eliminado

quarto estado (trem 3 aguarda):

tabela de tempos atualizada:

	$seção_1$	$seção_2$	$seção_3$	$seção_4$	$\mu_i(t_{i,j})$
$trem_1$	0.0	0.75	2.0	2.75	0.722
$trem_2$	3.0	2.0	1.0	-	1.0
$trem_3$	4.75	3.75	2.75	1.0	0.631

FA = 0.631

elimina primeiro estado

seleção do quarto estado, FA = 0.631

próximo conflito: não há , terminou.

3.6.7 - Integração de Métodos de Raciocínio e Busca

A utilização de algoritmos de busca para a solução de problemas reais, principalmente o A*, é cada vez mais crescente. Dentre os trabalhos que mais se destacam, podemos citar o método utilizado por (Y. Kobayashi, Y. Wada e T. Kiguchi [1986]) que empregam o algoritmo A* para dirigir a busca de uma rota ótima entre dois obstáculos situados em uma planta "layout".

A observação principal a respeito deste método é que o mesmo propõe um sistema de restrições de forma a restringir o espaço de estados (células) durante o procedimento de expansão (geração de estados sucessores). Outro comentário importante é que o método realiza uma inferência em uma base de conhecimento obtendo um conhecimento específico (conhecido como módulo de redução de conhecimento) que estabelece especificações para o planejamento de rotas. A seguir, este conhecimento é reduzido em um conhecimento primitivo na forma de restrições, índices de performance e critérios que são utilizados, posteriormente, na determinação e avaliação de caminhos (ligação entre células).

A utilização de uma base de conhecimento, com informações a

respeito do domínio do problema, para suportar uma busca dirigida no espaço de estados é de fundamental importância para a redução de complexidade de problemas reais. O desenvolvimento desta base para o problema de sequenciamento de trens será considerado, nos próximos capítulos.

Capítulo IV
Desenvolvimento de um Sistema Baseado em Conhecimento

4.1 - Viabilidade de Utilização em Sistemas de Planejamento e "Scheduling"

Neste capítulo, apresentaremos a descrição de um sistema baseado em conhecimento. Mais especificamente , de um sistema de produção combinado com a utilização de "frames" cuja implementação se baseia na filosofia de tipos abstratos de dados, extensões de tipos e abstração funcional.

Segundo H. Simon [1986] o principal papel de um sistema baseado em conhecimento é prover o conhecimento necessário a respeito do domínio do problema, permitindo a realização de uma busca heurística ou de um processo de escolha seletivo que conduza a boas soluções.

Neste sentido, o sistema foi projetado com a finalidade de suportar a realização de planejamento, determinando ações satisfatórias, em um tempo de resposta compatível com as funções do sistema inteligente voltado para o gerenciamento e controle do tráfego de trens.

Primeiramente, vamos abordar a viabilidade da utilização de sistemas baseados em conhecimento nas tarefas de planejamento e "scheduling", relacionadas ao gerenciamento e controle de trens, apresentando alguns sistemas bem sucedidos. Em seguida, vamos descrever as formas principais de representação do conhecimento e métodos de raciocínio, destacando o desenvolvimento de sistemas de produção e a utilização de "frames". Dando continuidade, apresentaremos uma descrição geral e detalhes de implementação de nosso sistema.

4.1.1 - Planejamento e "Scheduling"

No capítulo II, onde fizemos uma descrição do SIGT, verificamos a possibilidade de realizar o controle e gerenciamento de trens através do gerenciamento de plano de ações compreendendo um conjunto de funções de planejamento, distribuídas em níveis ou etapas. Dentre as funções mais importantes, estão a realização do planejamento com replanejamento, necessários a projeção e consequente reelaboração de planos de horários em tempo real.

No capítulo III, vimos que o problema de elaboração e reelaboração de planos de horários, a nível tático de planejamento, pode ser modelado como um problema especial de "job-shop scheduling" possuindo um conjunto de restrições estruturais (restrições de sequenciamento e precedência) e um conjunto de restrições flexíveis, representando os objetivos .

Observamos, também, que a nível operacional de planejamento, o problema de "scheduling" caracteriza-se como um problema mal-estruturado (H. Simon [1973]) necessitando, segundo M.S. Fox [1989], de uma filosofia nova, que integra um conjunto de técnicas de Inteligência Artificial favoráveis a solução do problema. Para tanto, apresentamos uma modelagem do problema, seguindo a metodologia clássica de Programação Matemática, e , em seguida, proporcionamos uma evolução na modelagem determinando um espaço de problema caracterizado por uma representação simbólica ou estado, um conjunto de operadores e uma função de avaliação , que permitiram a realização de uma busca heurística direcionada pelas restrições estruturais e flexíveis do problema, ou seja, orientada para a satisfação das restrições e realização dos objetivos .

Neste capítulo, apresentaremos o desenvolvimento de um sistema baseado em conhecimento cujo papel fundamental, é proporcionar a realização de uma busca inteligente, através da determinação de preferência em operadores, reduzindo o espaço de soluções viáveis do problema. Além desta função, o sistema baseado em conhecimento permite lidar com o aspecto dinâmico e complexo do problema de planejamento e "scheduling" quando envolvido com aplicações em tempo real.

Últimamente, os sistemas baseados em conhecimento tem sido aplicados com bastante frequência na solução de problemas de planejamento e "scheduling", principalmente, nos sistemas voltados para a produção de manufaturados.

4.1.2 - Aplicações Existentes

Para um estudo mais detalhado sobre a aplicação de técnicas de Inteligência Artificial em sistemas de planejamento e "scheduling",

recomendamos a leitura do trabalho de A. Kusiak e M. Chen[1987], onde são descritas as principais pesquisas e aplicações de sistemas especialistas em planejamento e "scheduling" na produção de manufaturados, bem como, os componentes principais destes sistemas, destacando as formas de representação de conhecimento e os métodos de raciocínio. Os autores observam, também, a importância da integração de técnicas de Inteligência Artificial e Pesquisa Operacional.

Entre os sistemas apresentados no trabalho de Kusiak e Chen, destacamos:

1 - Sistemas baseados em regras:

. G. Bruno, A. Elia e P. Laface [1986], voltado para a programação de tarefas em um sistema de manufaturados flexíveis. Desenvolvido em OPS5 e Fortran. Utiliza modelos de simulação para avaliação de sistemas especialistas;

. D. Ben-Arieh [1986], problema de "shop-floor scheduling". Desenvolvido em Prolog, Pascal e Slam-II. Utiliza algoritmos para cálculo de custos;

. P. Subramanyam e R. Askin [1986], problema de "job-shop schduling" em tempo real. Desenvolvido em Prolog e estruturado em três níveis de hierarquia ;

. M. Shaw e A. Whinston [1985], problema de controle e planejamento de manufaturados. Desenvolvido em LISP com arquitetura distribuída;

. P. Newman e K. Kempf [1985], programação de robos. Desenvolvido em Pascal como um modelo hierárquico de regras.

2 - Sistemas que utilizam "frames":

. M. S. Fox [1983], gerenciamento inteligente em um problema de "job-shop scheduling". Desenvolvido em SRL , "Schema Representation Language" . Utiliza estrutura hierárquica, modelos de simulação e tem como estratégia de controle busca direcionada por restrições;

. S. Shen e Y. Chang [1986], problema de "job-shop scheduling". Utiliza algoritmos de "scheduling";

. M. Shaw e A. Whinston [1986], problema de "job-shop

scheduling". Utiliza técnica hierárquica de solução de problemas;

. T. Morton e T. Smunt [1986], problemas de projeto, planejamento e "scheduling". Desenvolvido com estrutura hierárquica. Utiliza modelos de Pesquisa Operacional .

A descrição destes e de outros sistemas, serão consideradas em nosso trabalho, sempre que necessário. Ou seja, sempre que abordarmos alguma característica importante destes sistemas relacionada a estrutura e funcionalidade do SIGT.

Finalmente, queremos destacar no trabalho de Kusiak e Chen, sua parte conclusiva, que, a exemplo de nosso ponto de vista, sugere o desenvolvimento de sistemas para a solução de problemas mal-estruturados, observando a utilização de :

- sistemas baseados em regras;
- processo de busca direcionado pela satisfação de restrições;
- utilização de heurísticas;
- integração de modelos de simulação e
- integração de modelos de Pesquisa Operacional.

4.2 - Representação do Conhecimento

4.2.1 - Bases de Conhecimento

Sistemas baseados em conhecimento são programas de computador que representam de forma explícita e declarativa o conhecimento relativo a um domínio específico, envolvendo aspectos do conhecimento humano com o objetivo de executar tarefas, até então, restritas a especialistas.

O termo sistema especialista é utilizado como forma de enfatizar que o sistema trabalha como se fosse um especialista humano, considerando o aspecto específico do conhecimento.

O conhecimento em uma base é representado em uma linguagem formal e extraído por um interpretador que deve ser independente do domínio. Um mecanismo geral de interpretação extrai o conhecimento através de inferências, sobre um conjunto de fatos (axiomas), conduzindo

a uma importante conclusão ou a solução de um problema específico.

Em Inteligência Artificial, estudos que consideram a aplicação de métodos de raciocínio em representação e utilização do conhecimento se relacionam as formas como o conhecimento pode ser expressado e extraído de sua base. Portanto, veremos, a seguir, algumas formas básicas de representação do conhecimento, bem como, os processos de raciocínio associados.

4.2.2 - Objeto-Atributo-Valor

- Tripla Objeto-Atributo-Valor:

Uma maneira comum de representarmos informações sobre fatos e através do uso de triplas objeto-atributo-valores (O-A-V). Este esquema de representação é utilizado por inúmeras ferramentas voltadas para a construção de sistemas especialistas como M1 e S1 (Tecnowledge Inc.) que utilizam estratégias de encadeamento de regras para trás e OPS5 (Carnegie-Mellon University) uma linguagen para a programação de sistemas de produção com estratégia de encadeamento de regras para frente.

Uma tripla O-A-V fica caracterizada pelo uso de:

- objetos: que representam entidades conceituais ou físicas;
- atributos: que representam características gerais e propriedades, e
- valores: que são assumidos pelos atributos.

Neste caso, temos a existência de dois tipos de relacionamento entre os componentes de uma tripla O-A-V:

- objeto ---"has-a"---> atributo, indicando a posse de uma característica ou propriedade e
- atributo <---"is-a"--- valor, indicando a atribuição momentânea de um valor ao atributo.

Os objetos podem ser relatados hierarquicamente, formando uma árvore de O-A-V. Seguindo o trabalho de A.T. Bahill, P. Harris e E.

Senn[1988], apresentamos uma árvore de O-A-V voltada para a representação simplificada da taxonomia de trens em uma ferrovia, ver figura 34.

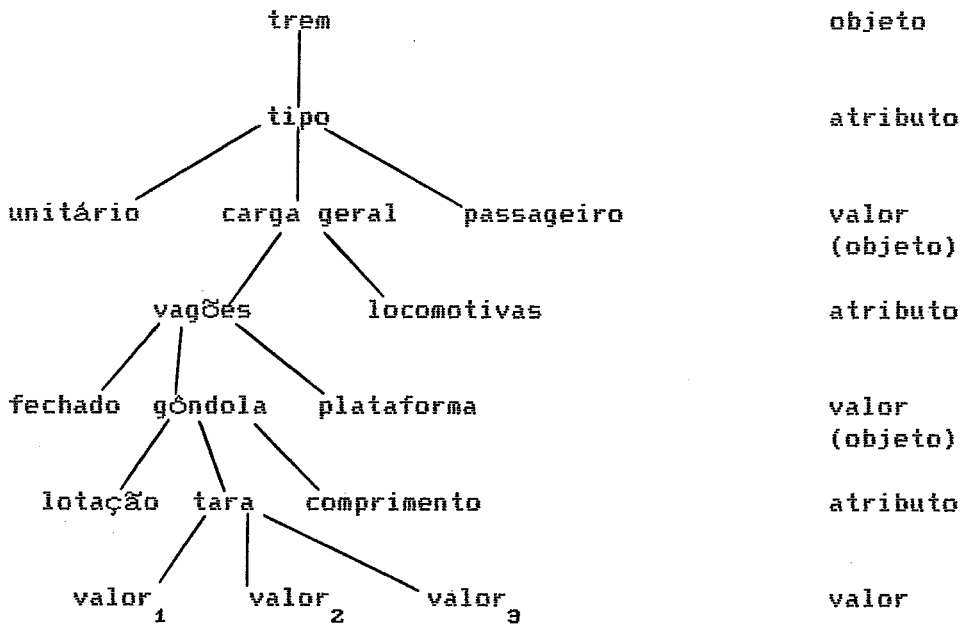


figura 34: taxonomia de trens em uma árvore O-A-V.

As triplas O-A-V, segundo P. Harmon e D. King [1985], podem, também, serem organizadas em uma rede, formando um caso especial de rede semântica. Entretanto, a forma mais comum de apresentação de uma tripla O-A-V, é através de sentenças isoladas na forma: " um objeto tem um atributo e seu atributo é um valor."

- Conhecimento Estático e Dinâmico:

As informações sobre atributos de um objeto podem alterar-se dinâmicamente. Ao processo de atribuir a um determinado atributo um valor específico denominamos instanciação. Em geral, consideramos os objetos e respectivos atributos informações estáticas de uma base de conhecimento. Ao passo que uma instância de alguma tripla O-A-V é considerada como uma informação dinâmica, definindo, no caso dos sistemas de produção, um elemento da memória de trabalho.

Se, em uma hierarquia de triplas O-A-V, atribuímos valores a objetos , todos os objetos e atributos situados em níveis inferiores da

hierarquia herdam estas informações.

- Pares Atributo-Valor:

Quando o sistema trabalha somente com um tipo de objeto, o mesmo pode ser suprimido e os fatos representados na forma de pares atributo-valor (A-V).

Estes sistemas, por não apresentarem múltiplos objetos não podem utilizar formas de relacionamento definidas por herança, ou desenvolver uma hierarquia de O-A-V.

Como maior exemplo da utilização de pares A-V podemos citar a linguagem de programação Prolog. Em Prolog (N.C. Rowe [1988]), predicados representam assertivas ou informações factuais,podendo se apresentarem nas seguintes formas:

. predicado tipo ou na forma A-V, contendo 1 argumento:

Ex: veículo(trem);

. predicado com propriedades, ou na forma O-A-V, contendo 2 argumentos:

Ex: tipo(carga-geral,trem), comprimento(valor$_9$,gôndola);

. predicado com relacionamento, também na forma O-A-V, contendo 2 argumentos:

Ex: parte-do(vagão,carga-geral).

Existem, também, na linguagem Prolog predicados do tipo funções, probabilidade e para a criação de registros de dados.

4.2.3 - Redes Semânticas

- Modelo Entidade-Relacionamento:

Uma rede semântica reflete a conectividade do conhecimento, sendo constituída, basicamente, de um conjunto de objetos conectados por arcos . Segundo P. Harmon e D. King [1985], não existem restrições absolutas em relação ao significado dos objetos e arcos que compõe uma rede semântica. Entretanto, seguindo alguns conceitos do modelo

entidade-relacionamento de P. Pin-Shan Chen [1976] podemos, classificar objetos como:

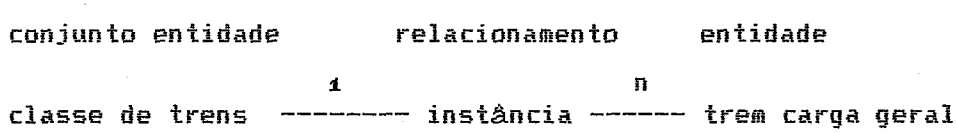
- entidades físicas;
- entidades conceituais e
- valores

Seguindo esta classificação, podemos considerar entidades individuais ou representativas de uma classe ou conjunto de objetos. Da mesma forma, os valores podem constituir, também, um conjunto específico , que formam os possíveis valores legais dos atributos das entidades.

Os arcos,no modelo entidade-relacionamento, de um modo geral, podem representar:

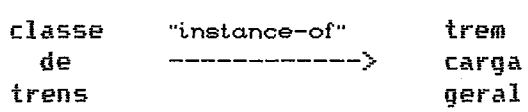
- relacionamento entre entidades;
- relacionamento entre entidades e valores .

No primeiro caso, relativo ao relacionamento entre entidades, podemos distinguir a representação do relacionamento classe e instância. Segundo o modelo de P. Pin-Shan Chen, devemos identificar uma informação semântica no tipo de relacionamento, definindo um mapeamento do tipo:



Ou seja, podemos criar "n" instâncias a patir de um conjunto entidade.

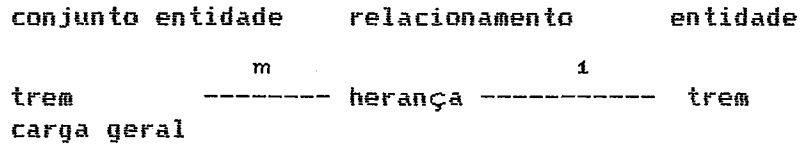
A instância de uma classe pode definir uma nova classe ou um novo conjunto entidade. Este relacionamento nas redes semânticas e representado pelo arco:



- Herança e Especialização:

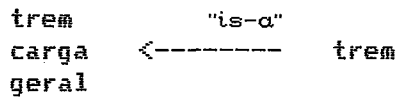
Outra forma comum de relacionamento entre entidades é aquela que

permite uma forma de herança . Neste caso, instâncias de classes ou entidades, definidas a partir de uma entidade-relacionamento apresentam um certo grau de dependência caracterizado pela herança de propriedades. Ou seja, podemos definir entidades trens a partir da relação:



permitindo, também, a ocorrência de herança múltipla a partir de diferentes conjuntos entidades.

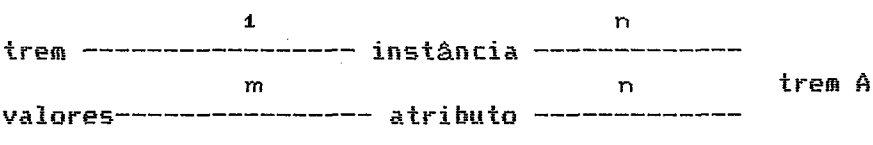
Na notação de redes semânticas, este relacionamento é expresso pelo arco:



Podemos observar que ao contrário do mapeamento de instância, este mapeamento é da forma m:1. Ou seja, cada trem pode ser definido a partir de múltiplas heranças herdadas de diferentes classes, formando uma nova classe única.

- Atributos:

No exemplo anterior, dizemos que a entidade trem especializa o conjunto entidade trem carga geral, através da incorporação de propriedades adicionais . Neste caso, podemos aplicar, novamente, o relacionamento "instance of", definindo objetos ou trens específicos com seus respectivos valores dos atributos adicionais:



Neste exemplo, nos referímos a segunda forma de relacionamento que caracteriza a relação entre entidades (objetos) e respectivos valores ou entidades conceituais (eventos, ações). Este

relacionamento é denominado atributo, podendo ser do tipo:

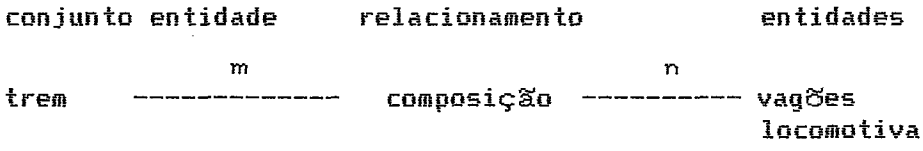
- denotativo (nome);
- casual (causado por);
- condicional (condição é) e
- descritivo .

Os atributos, segundo P. Pin-Shan Chen, podem ser definidos como mapeamentos de um conjunto entidade (E_i) ou um conjunto relacionamento (R_i), em um conjunto de valores (V_i) ou em um produto cartesiano de conjunto de valores, ou seja:

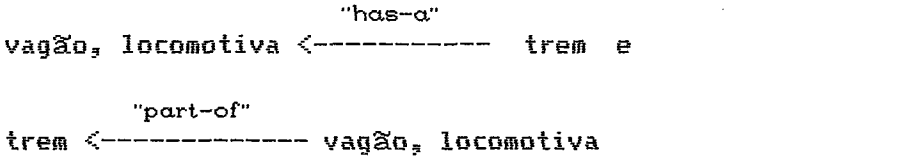
$$f : E_i \text{ ou } R_i \longrightarrow V_i \text{ ou } V_{1i} \times \ldots \times V_{ni}$$

- Composição:

Finalmente, podemos destacar outra forma de relacionamento, existente em uma rede semântica, que completa as duas dimensões de organização do conhecimento , caracterizadas pela especialização e composição. Para representar a composição, cada entidade ou objeto deve possuir um conjunto de regras de composição que especificam os componentes ou as legítimas combinações de partes de uma classe. Ou seja:



Na notação de redes semânticas, representamos este relacionamento pelos arcos:



As restrições de composição existentes para a definição de objetos e entidades formam uma hierarquia denominada hierarquia "part-of" (H. Levesque e J. Mylopoulos [1975]) ou hierarquia de composição (A. Mackworth e W. Havens [1981]) . Já as restrições de

especializações definem uma forma de hierarquia, muito comum, denominada hierarquia "is-a" que reflete a taxonomia de heranças.

4.2.4 - "Schemas"

Willian Havens [1985] apresenta um trabalho onde descreve um modelo ou uma base de conhecimento constituída de "schemas" para a representação de trens em uma ferrovia, utilizando composição e especialização para expressar o relacionamento estrutural. Havens descreve um conjunto de "schemas" definindo: um conjunto de regras de composição, um conjunto de componentes, um conjunto de restrições que determinam especializações e composições de trens e um conjunto de sub-classes que determinam uma hierarquia de especializações de objetos.

Portanto, um "schema" S_i da base de conhecimento tem a seguinte seguinte representação:

$S_i = (P_i, B_i, C_i, D_i)$

Exemplo:

trem:

P = {p1,p2} (regras de composição)

B = {tração,vagão,caboose} (componentes ou "schemas" para classes)

C = {R [{trem,blococarro}],R [{trem,tração}],R [{trem,caboose}],
1,trem 2,trem 3,trem
(restrições de composição)

R [{trem,longdist}], R [{trem,curtdist}]}
longdist,trem curtdist,trem
(restrições de especialização)

D = {longdist, curtdist} (conjunto de sub-classes)

longdist (longa distância):

P = 0;

B = 0;

C = {R [{longdist,carro}],[R [{longdist,tração}],
1,longdist 2,longdist
R [{longdist,caboose}]}
3,longdist
(restrições de composição)

D = {passageiro, carga} (conjunto de sub-classes)

blococarro:

P = {p3,p4} (componentes)

B = {carro, blococarro}

C = {R [{blococarro,carro}], R [{blococarro,carro,blococarro}]}
1,blococarro 2,blococarro
(restrições de composição)

D = {blococarga, blocopassageiro, blocomixto}
(conjunto de sub-classe)

carro:

P = 0

B = 0

C = {R [{carro,carrocarga}], R [{carro, carropassageiro}]}
carro,carrocarga carro,carropassageiro
(restrições de especialização)

D = {carrocarga, carropassageiro} (conjunto de sub-classes)

carrocarga:

P = 0

B = 0

C = 0

D = {plataforma, gondola, fechado, tanque} (conjunto de sub-classes)

A hierarquia de especializações definida para a representação dos "schemas" acima, tem a forma, ver figura 35.

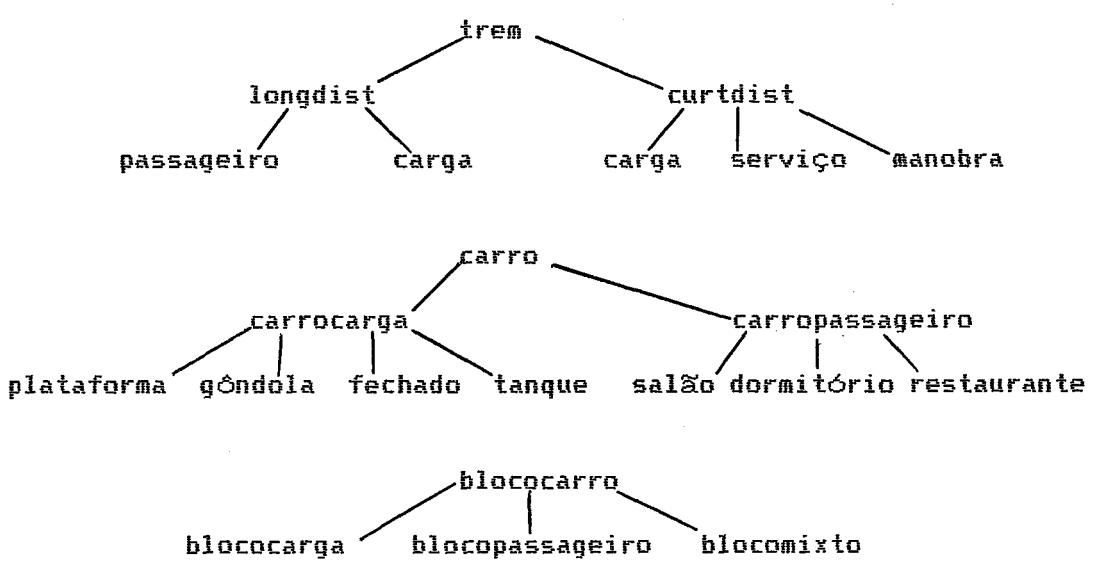


figura 35: hierarquia de especializações.

A hierarquia de composições tem a forma, ver figura 36.

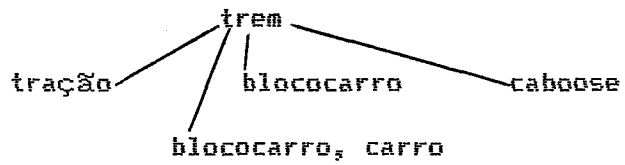


figura 36: hierarquia de composições.

Desta forma, é possível determinar instâncias de "schemas" (protótipos que representam uma classe geral de objetos) através da execução das regras de composição e especialização de uma forma semelhante as linguagens de programação logica. Ou seja, o controle da execução segue a estratégia top-down com realização de "backtraking" nos casos de falha, até atingir o objetivo estabelecido (determinação de uma instância completa).

O sistema de representação de conhecimento apresentado por Havens é ideal para problemas de reconhecimento, já que é possível verificarmos a validade de hipóteses pelo método de refutação, a exemplo da linguagem PROLOG.

4.2.5 - Regras de Produção

- Organização:

Uma das formas mais naturais de representação do conhecimento, ou do relacionamento do conhecimento, relativo a técnicas de solução de problemas, é através de um conjunto de regras na forma: Se condição Então ação. Estas regras, segundo Hebert Simon [1981], são consideradas unidades independentes de conhecimento ou "chunks", determinando uma grande facilidade na representação e manipulação do mesmo.

Os sistemas baseados em conhecimento que contem conhecimento armazenado na forma de regras, são conhecidos como sistemas baseados em regras ou sistemas de produção. Estes sistemas, além de representarem o conhecimento, constituem uma técnica bastante eficiente na solução de problemas.

Um sistema de produção consiste, basicamente,de três componentes principais, segundo F. Hayes-Roth e outros [1983]:

1 - Uma base de dados, comumente chamada de memória de trabalho, que contém elementos que representam situações particulares de um conhecimento de curto prazo (dinâmico);

2 - Uma base de regras ou memória de produção que representam o conhecimento especialista, de longo prazo, associado e dependente do domínio do problema.

3 - Um interpretador, também denominado mecanismo de inferência, que examina, reconhece e provoca a exucução das regras, de forma cíclica, provocando alterações sucessivas no estado da memória de trabalho.

- Interpretação, Controle e Execução:

Mais precisamente, uma regra assume a seguinte forma:

Se condição$_1$ E condição$_2$ ... E condição$_n$ Então ação

Caso as condições do lado direito ou antecedente da regra se refiram a uma proposição verdadeira, observando-se os conectivos lógicos, a mesma podera ser candidata a ativação, provocando, neste caso, a execução da ação desejada , ou execução do consequente.

A execução de alguma regra, proporciona a modificação, criação ou deleção de algum elemento da memória de trabalho. Desta forma, o conhecimento associado a memória de trabalho tem um aspecto dinâmico, condicionado a execução das regras de produção.

É muito comum, também, estar associado as condições de uma regra, informações relativas a um conhecimento estático representado por estruturas de armazenamento denominadas "frames". Este conhecimento é, essencialmente, declarativo e relacionado as informações factuais, podendo, em alguns casos sofrer alterações.

Para verificarmos se a condição de uma regra é uma proposição verdadeira, realizamos a interpretação da regra. A forma mais comum de interpretação, consiste em verificarmos se ocorreu a instanciação do conjunto de componentes na forma O-A-V que compõem a condição da regra.

Na forma mais simples as componentes da condição da regra se referem a padrões ou valores legais da memória de trabalho. Neste caso, a ocorrência de ligação ou "match" entre os componentes das regras e os elementos da memória de trabalho é verificada pelo casamento de padrões.

Quando completa o ciclo de interpretação, caso ocorra mais de uma regra aplicável, surge uma situação de conflito que é resolvida pela utilização de alguma estratégia adotada.

É interessante observamos que no estilo de programação convencional, as estruturas de controle da linguagem (procedimental) determinam a ordem de execução do programa (forma determinística). Entretanto, nos sistemas de produção a ordem de exucução é determinada pelos mecanismos que regulam a interpretação e seleção de regras determinando uma conduta não determinística e, segundo P. Wegner[1986] , um esquema de classificação progressiva. Na programação orientada para objetos a forma básica de controle e centrada na troca de mensagens ou trensferência de controle entre processos (co-rotinas).

Quando o sistema trabalha segundo a interpretação de uma base de dados, gerando uma árvore de conclusões cujas ramificações dependem das ações provocadas pelas regras, dizemos que o raciocínio do mesmo é encadeado para a frente ou a execução de regras é orientada pelos dados. Estes sistemas são muito utilizados em problemas de planejamento, projeto, diagnóstico e monitoração.

- Raciocínio e Inferência:

A estratégia de inferência utilizada por estes sistemas é a aplicação do modus-ponens. Ou seja, se tivermos uma regra do tipo:

Se (A) Então (B) ,

e soubermos da verdade da condição (A),então é válida a execução de (B), pois por dedução natural:

Se (A) Então (B) $\Rightarrow$ $\neg(A)\vee(B)$

$\neg(A)\vee(B) \wedge (A) \Rightarrow$ (B) (hipótese e eliminação)

Uma das vantagens da utilização de sistemas baseados em regras é a sua capacidade de justificar as conclusões. Para tanto devemos

estabelecer a cadeia de inferência ou caminho, relativo a execução das regras, transcrevendo o antecedente e consequente das mesmas na forma textual.

4.2.6 - " Frames "

- Organização e Estruturação do Conhecimento:

M. Minsky [1975] sugeriu que o conhecimento de domínios deveria ser representado como um conjunto de "frames", correspondendo a classes de objetos com seus respectivos relacionamentos.

Desta forma, uma "frame" deveria conter, basicamente, um conjunto de "slots" que representassem o relacionamento entre objetos e o conhecimento declarativo, compreendendo:

- relacionamento entre "frames";
- predicados com um e vários argumentos e
- procedimentos associados.

Entre os tipos de relacionamentos existentes entre as "frames" destacam-se a capacidade de herança e instanciação automática. Os predicados, em geral, representam fatos, atributos e propriedades característicos da classe de objetos e independentes do domínio. Finalmente, os procedimentos associados determinam a conduta e representam o conhecimento imperativo, funcional e dependente do domínio.

Segundo R. Fikes e T. Kehler [1985], as linguagens para construção de "frames" permitem uma representação estrutural concisa do conhecimento, bem como de seu relacionamento, através da técnica básica de organização denominada especialização, permitindo, também, forte poder de expressividade, bem como, acessibilidade desejável, através do uso de mecanismos de inferência que exploram as características estruturais das mesmas. Para Fikes e Kehler, sistemas de "frames" são usuais para a representação de conhecimento na forma declarativa, através de uma linguagem de cálculo de predicados, tornando-se adequados para o uso em problemas de reconhecimento, através da verificação de ligações ("matching"), sendo bastante útil em tarefas de diagnóstico e

classificação. Conforme observações feitas por R. J. Brachman [1988], P. Szolovits [1986] e o próprio Minsky [1975], esta era a concepção inicial destes sistemas de representação do conhecimento.

Dando sequência ao trabalho de Minsky, tem aparecido um grande número de sistemas ou linguagens de "frames", com maior ênfase na estrutura de representação e nas formas de relacionamento entre "frames" , divergindo, portanto, da concepção inicial. Nestes sistemas, uma "frame" é vista como uma representação estruturada de um objeto ou de uma classe de objetos, contendo formas de conduta e um conjunto de valores "default". Entre os sistemas mais importantes, podemos destacar as linguagens KRL, UNITS e KL-ONE.

As informações providas pelas "frames" podem ser consideradas como um "banco de dados" alternativo, relativo a parte estática das bases de conhecimento, ficando o controle do conhecimento e as formas de reciocínio para outros componentes desta base. Esta nova visão reforça a utilização combinada de sistemas de "frames" e sistemas de produção.

Conforme observações de Fikes e Kehler [1985], as regras de produção são inadequadas para definir o relacionamento estático entre objetos e não detem mecanismos automáticos de inferência. Entretanto, este é o ponto forte dos sistemas baseados em "frames". Por outro lado, os sistemas de produção possuem como ponto forte a representação do conhecimento dependente do domínio e a possibilidade de realização do controle de forma racional e objetiva. Desta forma, tem-se desenvolvido, recentemente, ferramentas híbridas de programação de sistemas em Inteligência Artificial, que combinam a utilização de "frames" e produções. Entre estas novas ferramentas podemos destacar as linguagens LOOPS, KEE e SRL . Esta, também, e a concepção adequada ao sistema baseado em conhecimento desenvolvido para o SIGT, o qual será descrito, no próximo capítulo , com maiores detalhes .

Conjuntamente, abordaremos algumas questões relativas as características estruturais de "frames", incluindo descrição de "slots", tipos de relacionamento e facilidades para a representação do conhecimento procedimental ou dependente do domínio. Por hora, vamos nos ater a discussão sobre a utilização ou aplicação de "frames", segundo:

. uma base estática para sistemas de produção;

. um modelo de gerenciamento de módulos ou sistemas de regras, e
. um sistema para a monitoração de eventos externos.

- Base Estática para Sistemas de Produção:

A representação de fatos ou axiomas em uma base de conhecimento é essencial para o funcionamento de sistemas de produção. Posteriormente, veremos que a representação do conhecimento estático nestes sistemas, incluindo conhecimento factual, na forma de redes de triplas O-A-V ou listas de pares A-V, ou seja, como elementos da memória de trabalho é extremamente desaconselhável. Desta forma, algumas arquiteturas de sistemas baseados em regras, como por exemplo o sistema ISIS (J. McDermott e MS. Fox [1986]) empregam o desenvolvimento de banco de dados, como forma de armazenamento do conhecimento estático e factual do domínio.

Dada a versatilidade e potencialidade das "frames" para representar este conhecimento, sugerimos a utilização de suas informações nos componentes das regras de produção. Desta forma, o processo de raciocínio (interpretação de condições) é realizado automaticamente através da comparação lógica de valores ativos referentes aos "slots" de "frames". Na implementação realizada para o SIGT é possível estabelecer a visibilidade dos "slots" de "frames" para cada módulo de regras ou conjunto de produções.

- Modelo de Gerenciamento de Módulos de Regras:

No caso anterior, vimos a utilização de "frames" em módulos de regras. Agora, vamos ver a utilização de módulos de regras ou conjunto de produções em "frames".

Na linguagem de "frames" KEE, uma regra pode ser representada por uma fórmula que utiliza uma linguagem de cálculo de predicados para representar as premissas (condições) e assertivas (ações). A linguagem de cálculo de predicados possue como literais formas de relacionamento que podem serem representadas pela linguagem de "frames". Esta linguagem permite, também, que alguma função externa computável em LISP, seja utilizada como predicado. Desta forma, uma parte significante do raciocínio, necessário a interpretação das regras, é

realizada pela "frame". A linguagem KEE permite, também, que um conjunto de regras sejam agrupadas e associadas a uma classe que, por sua vez, é associada a um "slot" da "frame".

A informação contida neste "slot" pode ser acessada de duas formas:

. através de métodos ou resposta a mensagens. Neste caso, sempre que uma mensagem é enviada a "frame", o conjunto de regras associado a mensagem é, imediatamente, invocado;

. através de valores ativos. Neste caso, sempre que o valor do "slot" é requerido, o conjunto de regras ou função associado é, também, imediatamente invocado.

Como qualquer outra informação associada a "slots", módulos de regras podem ser herdados automaticamente por outras "frames" que possuam um relacionamento de herança.

O uso de regras ou classe de regras integrado a "frames" na linguagem KEE expressa, essencialmente, o relacionamento condicional entre informações, sendo de grande aplicação em sistemas de diagnóstico voltados para a classificação de situações. O encadeamento de raciocínio é possível através da atualização automática de informações e consequente instanciação de regras, não sendo utilizado, para tanto, a memória de trabalho.

No sistema SIGT, apresentamos uma abordagem diferente, semelhante a destacada anteriormente, onde o encadeamento de regras é proporcionado pela evolução dinâmica dos elementos da memória de trabalho. Desta forma, em nossa implementação, não permitimos que o consequente das regras provoquem alterações nas informações exportadas pelas "frames", mas, somente, em valores desta memória.

Finalmente, vale ressaltar que na linguagem KEE é permitido a verificação de hipóteses através da estratégia de caminhamento ou encadeamento de regras para trás. Caso alguma hipótese tenha conclusão verdadeira, a mesma é, automaticamente, incluída na base de conhecimento. É possível estabelecer uma forma de controle do conhecimento através da organização e indexação de módulos de regras em "frames".

- Sistema para a Monitoração de Eventos Externos:

A linguagem KEE permite que "slots" associados a métodos respondam a mensagens como alarmes. Ou seja, existe um "slot" que sinaliza sempre que uma regra ou classe de regras produz algum valor específico.

Alternativamente, qualquer "slot" associado a um valor ativo pode ser monitorado quanto a possíveis trocas de valores "default" ou variações segundo intervalos aceitáveis. Podemos, também, considerar a possibilidade de "slots" associados a valores ativos determinarem a chamada de funções, ou métodos invocarem módulos de regras em resposta a mensagens, sempre que alguma condição imposta for satisfeita.

Uma interessante aplicação de sistemas de "frames" que gerenciam módulos de regras se refere ao desenvolvimento do sistema Star-Plan (J.C. Ferguson, R. Siemens e R.E. Wagner [1985]). Este sistema, consiste de uma linguagem de "frames" que descreve as partes de um sistema de satélites de comunicação, definindo um conjunto de protótipos associados a métodos que descrevem o relacionamento entre partes, bem como a conduta particular de cada parte.

Existem, no sistema Star-Plan, dois tipos fundamentais de classe: guardiãs, responsáveis pela observação e diagnóstico; e monitores, responsáveis pela análise e reparo.

Desta forma, quando algum tipo de problema ocorre em uma parte do satélite, é enviada uma mensagem que acorda o guardiã associado. O guardiã, por sua vêz, invoca uma classe de regras de diagnóstico, com encadeamento para frente, que especifica o tipo de problema. Quando o guardiã determina o tipo de problema, o mesmo envia um sinal para o monitor responsável pela solução do tipo de problema encontrado. Desta forma, o monitor invoca um conjunto de regras de análise que, através da estratégia de caminhamento para trás, testa hipóteses específicas sobre o problema e faz recomendações de reparo ao operador. É importante ressaltar que, as classes de monitores são criadas e destruídas dinâmicamente, liberando a memória da CPU.

4.3 - Sistemas Baseados em Regras

4.3.1 - Propriedades e Evolução

Segundo F. Hayes-Roth [1985] as propriedades mais importantes dos sistemas de produção são:

- incorporam conhecimento prático e humano na solução de problemas;
- conduta não-determinística, ou seja, adaptativamente determina a melhor sequência de regras a executar. A cadeia de inferência é construida dinâmicamente;
- voltado para a solução de problemas complexos e mal-estruturados;
- base de conhecimento modular, independente e homogênea;
- facilidade para a expansão e manutenção do conhecimento;
- modelagem da memória segundo o sistema humano de processamento da informação (A. Newell e H. Simon [1972]), e
- facilidade para explanação de resultados.

Estas propriedades baseiam-se, principalmente, na capacidade das regras de executarem as seguintes funções:

- decomposição paralela da conduta transiente (sucessão de estados da memória de trabalho) do programa, simplificando o controle e execução de instruções;
- simulação de dedução e raciocínio através de comparações lógicas e utilização do "modus-ponens";
- simulação de tomada de decisões determinando uma conduta objetiva, atraves da incorporação de heurísticas em regras.

Hayes-Roth analiza, também, o fato de que a evolução dos sistemas baseados em regras é um esforço conjunto no desenvolvimento e emprego de conceitos e técnicas de psicologia, matemática e teoria da computação, no intuito de tentar simular o processo de pensamento humano e utilizar o conhecimento de especialistas.

Desta forma, é apresentada a interconexão e consequente evolução destes sistemas associados a psicologia, teoria de computação e áreas de

aplicação. Particularmente, na área da teoria de computação temos a seguinte evolução, ver figura 37.

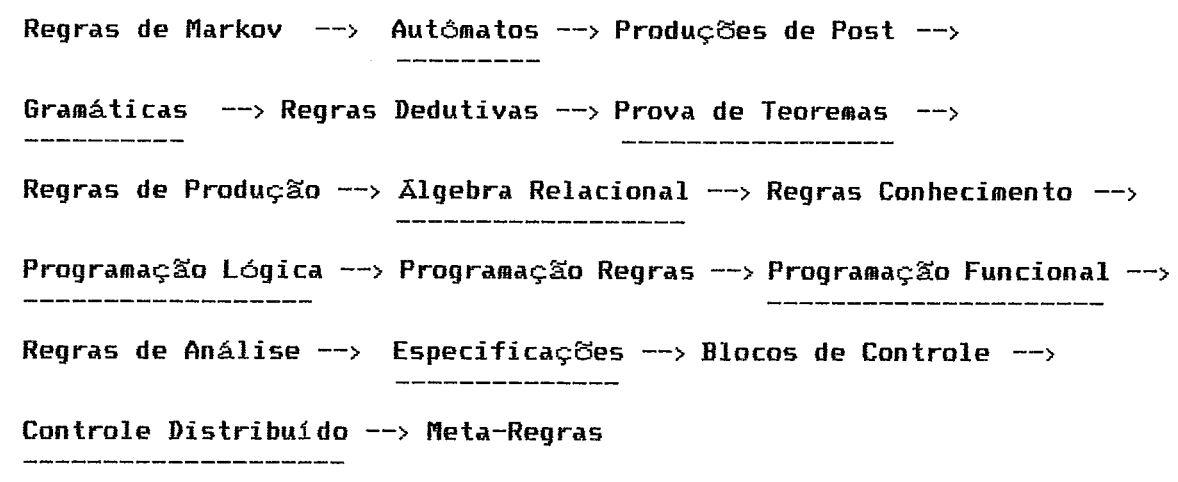


figura 37: interconexão entre sistemas baseados em regras e teoria de computação.

Inicialmente, Regras de Markov permitiam uma forma simples de definição de processos estocásticos com regras de probabilidade que possibilitaram o mapeamento e respectiva análise de conduta de estados em sistemas markovianos. Em seguida, Emil Post [1936] sugeriu máquinas abstratas que tinham a capacidade de computar todas as funções computáveis (Tese de Church).

Posteriormente, foram desenvolvidas prova de teoremas e uma linguagem para o cálculo de predicados que permitiram a dedução automática e verificação de formulas através do método de resolução.

Estes estudos, levaram ao desenvolvimento de sistemas de produção com regras na forma Se condição Então ação. Inicialmente, estas produções enfatizavam a incorporação de unidades de conhecimento. Com o sucesso do uso destas regras em sistemas especialistas, surgiram os sistemas baseados em regras definindo um novo paradígma de programação. A partir daí, o conhecimento incorporado as regras evoluiu, adquirindo, além do conhecimento analítico e do controle dedutivo, formas de controle do conhecimento a respeito de como atingir a melhor eficiência na solução de problemas. Este controle de conhecimento incluiu a construção de blocos de controle, meta-regras e outras formas de controle procedimental característico de linguagens procedimentais de alto nível.

4.3.2 - Controle do Conhecimento

- Sistema de Produção Nao-Determinístico:

Um sistema de produção não-determinístico é uma extensão direta de um sistema de produção de Post (Emil Post [1943]), consistindo, basicamente, de um conjunto de produções que alteram, sucessivamente, elementos de uma base de dados ou memória de trabalho, determinando uma sequência de transição de estados. Ou seja, segundo M.P. Georgeff[1982], um sistema de produção é uma máquina não-determinística (a escolha de caminhos é arbitrária e força a realização de "backtracking" em casos de falha) que descreve sequências de estados parciais de uma base de dados determinando uma relação entre estes estados.

Nestes sistemas, o processo de inferência consiste na realização do ciclo reconhecimento-ação com encadeamento de regras ou direção de avaliação para frente, selecionando um conjunto de regras que formarão o conjunto conflito em cada estágio de execução. Este procedimento é não-deterministico, e, para que possa ser implementado, necessita de uma estrátegia de ordenamento e seleção de regras a executar.

Os processos de seleção de regras consideram a possibilidade de ligações parciais nos componentes das regras, interferência de condições que definem a utilidade de aplicação de produções, conhecimento "a priori" do efeito destas aplicações na base de dados e utilização de estratégias que determinam um conjunto de regras ativas.

Uma vez, determinado o conjunto conflito, é necessário estabelecer um procedimento de escolha, ou uma estratégia de solução de conflitos. Entre as estratégias mais comumente utilizadas, podemos destacar, seguindo a ordem de preferência:

1 - as regras que não foram executadas recentemente tem preferência sobre outras regras;

2 - caso persista o conflito, de preferência as regras cujos antecedentes façam referências à elementos da memória de trabalho mais recentemente modificados;

3 - caso persista o conflito, de preferência as regras com maior número de componentes no antecedente e, finalmente,

4 - faça uma escolha aleatória.

Nos sistemas realmente não-determinísticos as estratégias de busca descritas não devem influenciar o tipo de solução encontrada. Entretanto, a maioria das arquiteturas existentes para sistemas de produção não respeita esta característica. Segundo M.P. Georgeff [1982], estes sistemas apresentam um esquema de solução de conflitos do tipo irrevogável, pois a melhor instância de regra que atende as estratégias estabelecidas, é selecionada a cada iteração, descartando-se as demais. Ainda, segundo Georgeff, estes sistemas de produção são vistos, mais precisamente, como generalizações de algorítmos de Markov[1961] do que como sistemas de Post. Desta forma, o tipo de solução encontrada, dependerá da estratégia de busca utilizada; e, devido a inexistência de "backtracking", é construido, somente, um caminho solução que leva ao estado final desejado.

Em seu trabalho, Georgeff considera o estabelecimento de controle através da estratégia de solução de conflitos deficiente, e propõe uma nova forma de controle baseada na especificação de uma linguagem de controle, determinando um sistema de produção controlável.

Como observado por G. T. Vesonder [1988], o contraste marcante entre os sistemas baseados em regras e os sistemas procedimentais é caracterizado pela forma de representação e pelo controle do conhecimento necessário a solução de problemas.

Nos sistemas de produção o conhecimento fica representado, explícitamente, na forma de regras e o controle é efetuado de forma implícita pelo mecanismo de inferência que determina uma estratégia de busca. Por outro lado, nas linguagens procedimentais, o conhecimento é representado de forma implícita (estrutura de dados + algoritmos) e o controle do mesmo é efetuado, explicitamente, através das estruturas de controle do programa. Vesonder ressalta, também , que a grande modularidade e indepêndencia do conhecimento representado em produções é prejudicial, no sentido de expressar o conhecimento dependente e funcional na forma de sub-rotinas ou procedimentos.

Neste sentido, tem-se proposto mecanismos de controle adicionais aos sistemas baseados em regras, com o intuíto de extender o controle do conhecimento além da tradicional estratégia de solução de conflitos durante o processo de inferência. Entre as propostas mais importantes,

vamos destacar, a seguir, o uso de meta-regras, controle procedural e abstração de espaços.

- Meta-regras:

Meta-regras são regras que contém uma forma de meta-conhecimento (conhecimento sobre o próprio conhecimento e formas de raciocínio) que permite especificar o controle na execução de regras. Como exemplo de implementação vamos descrever como D. Rosenthal [1985] propõe a inclusão de meta-regras na linguagem OPS5, mantendo seu estilo tradicional de programação.

Em seu trabalho são consideradas duas propostas básicas:

1- Através da representação de regras que determinam a prioridade de uma regra mais importante sobre um conjunto conflito de regras. Para tanto, é necessário a construção de uma nova ação no lado direito da regra e de um conjunto pré-definido de elementos da memória de trabalho, que representam o conjunto de regras instanciadas. Desta forma, uma meta-regra pode referenciar diretamente instâncias de regras pelo nome.

Exemplo (Rosenthal [1985]):

```
( meta-regra
{ < regra_mais_importante>
  ( conjunto_conflito
   ^tem_nome regra_nome1 ) }
  ( conjunto_conflito
   ^tem_nome regra_nome2 )
  ( conjunto_conflito
   ^tem_nome regra_nome3 )
-->
  (execute < regra_mais_importante >) )
```

Neste exemplo, a meta-regra determina a execução da regra de nome regra_nome1 quando a mesma se encontra no conjunto conflito com as regras de nome regra_nome2 e regra_nome3;

2 - Alterando, explícitamente, a estratégia de solução de conflitos através de uma rotina externa que possui um ponteiro para o conjunto conflito e uma variável global que preserva um ponteiro para a regra mais importante. Desta forma, até que uma meta-regra seja chamada

a variável preserva o ponteiro para NIL. Assim que a meta-regra é chamada, a ação no lado direito provoca a escolha da regra mais importante independentemente da estratégia de solução de conflitos.

Exemplo (Rosenthal [1985]):

```
( meta-regra
{ < regra_mais_importante>
  ( conjunto_conflito
   ^tem_nome regra_nome1 ) }
  ( conjunto_conflito
   ^tem_nome regra_nome2 )
  ( conjunto_conflito
   ^tem_nome regra_nome3 )
-->
  (faça  regra_a_executar < regra_mais_importante >) )
```

A grande desvantagem da linguagem OPS5 e da proposta de Rosenthal é que o uso de meta-regras fica restrito ao controle da execução de regras individuais, não suportanto a formação de blocos de controle na forma de abstrações funcionais. Um exemplo de linguagem que admite este processo de abstração é a ferramenta M.1 (Tecnowledge Inc.) através do uso de regras de controle que determinam a ativação de sequência de regras (W.B. Rauch-Hindin [1986]).

R. Davis [1980] considera o uso de meta-regras como um processo de refinamento ou reordenamento das regras selecionáveis para execução. Este conhecimento de controle, ou meta-conhecimento, é representado explícitamente, e organizado da mesma forma que o conhecimento associado ao domínio. Segundo Davis, o uso de meta-regras estabelece estratégias, com base em informações, no sentido de invocar, seletivamente, a fonte de conhecimento (regra ou conjunto de regras) a ser aplicada.

Neste sentido, segundo R. Davis, meta-regras são regras que possuem conhecimento a respeito de outras regras, no sentido de tornar uma fonte de conhecimento, em determinado contexto, preferível sobre outra.

- Controle Procedimental:

Conforme visto anteriormente, Georgeff [1982] propõe a construção de uma linguagem de controle procedimental para restringir, a cada fase de execução, o conjunto de produções a um sub-conjunto

específico, denominado conjunto ativo. Esta linguagem é independente da estratégia de busca adotada pelo mecanismo de inferência. Desta forma , o ciclo reconhecimento-ação toma a forma:

```
Repita

  Determine o conjunto de produções ativo ;
  ( controle )

  Determine o conjunto conflito;
  ( regras instanciadas )

  Selecione a produção;
  ( estratégia de solução de conflitos )

  Execute a produção selecionada;
  ( ação )

Até condição_de_parada;
```

As formas de controle especificadas pela linguagem podem variar de acordo com as formas usuais de controle das linguagens procedimentais.

Como exemplo, uma linguagem de controle regular ou do tipo 3, permite a realização do "loop" condicional através da execução sucessiva de um estado (conjunto ativo de produções) e posterior transferência a outro estado (outro conjunto ativo de produções). Este mecanismo de controle é fundamental no sentido de permitir o particionamento do sistemas de produção em sub-sistemas.

Outro exemplo seria estabelecer uma linguagem de controle livre de contexto ou do tipo 2, que permitiria a especificação de procedimentos recursivos e modulos de produções característicos das linguagens de programação de alto nível. Desta forma, fica possível estabeler uma organização hierárquica em sistemas de produção. Muito embora, a organização de modulos de regras provoque o agrupamento e consequente sequenciamento de produções, o esquema de busca ou avaliação sugerido para cada modulo é independente e livre, no sentido de prover nova forma de controle local.

- Abstração de Espaços:

Earl D. Sacerdoti [1974] apresenta uma forma de decomposição do espaço do problema em uma hierarquia de abstrações de espaços através da introdução sucessiva de níveis de detalhe. Este esquema de decomposição

permite a realização de uma técnica de planejamento hierárquico e incremental, com conduta top-down, aumentando, de forma significativa, a capacidade de solução de problemas.

Sacerdoti estabelece, com esta nova forma de representação, um solucionador de problemas denominado ABSTRIPS (abstração baseada em STRIPS),que utiliza a representação do domínio do problema proporcionada por STRIPS. O modelo STRIPS (R.E. Fikes e N.J. Nilsson [1971]) consiste de um conjunto de formulas bem formadas, wffs, na linguagem de cálculo de predicados, representando fatos e um conjunto de operadores descritivos que provocam transformações no domínio. Cada operador é representado por um conjunto de pré-condições e por duas listas que descrevem as modificações ocorridas na base de fatos decorrentes da aplicação do operador.

A estratégia de controle fundamental se baseia na técnica de redução de diferencas ou "means-ends analysis", no sentido de tornar a wff que representa o objetivo, verdadeira. Como a aplicação de um operador depende, essencialmente, de cada literal nas pré-condições existentes, ABSTRIPS define uma abstração de espaços atribuindo a cada literal um valor de criticalidade. Desta forma, é definida uma hierarquia onde os literais com valores mais críticos formarão um espaço de abstração mais alto, acumulando, a cada decréscimo do nível de criticalidade , níveis de detalhe representados pela inclusão de novos literais, com valores críticos mais baixos.

Esta técnica pode ser utilizada da mesma forma, em sistemas de produção, através do uso de parâmetros de criticalidade nos componentes das regras.

4.4 - Arquitetura de um Sistema Baseado em Conhecimento

4.4.1 - Considerações Iniciais

A utilização de sistemas especialistas ou sistemas baseados em conhecimento em tempo real tem aumentado em importância no campo de aplicações de técnicas de Inteligência Artificial, notadamente nos setores de róbotica, comunicações, projetos de defesa estratégica e controle de processos. Estes sistemas tem como paradígma ou proposta básica a solução de problemas relacionados as tarefas de interpretação, diagnóstico, prescrição, projeto, planejamento com replanejamento e monitoração. Segundo T.J. Laffey e outros [1988] a principal característica que define um sistema em tempo real é a capacidade do sistema em garantir respostas em um intervalo de tempo compatível com a aplicação afim.

Tradicionalmente, os sistemas baseados em conhecimento são voltados para aplicações onde o domínio é considerado estático e os tempos de resposta flexíveis quando comparados as exigências dos sistemas em tempo real. Estas aplicações são caracterizadas, principalmente, pelas tarefas de diagnóstico e projeto, podendo citar, como exemplo, os sistemas especialistas mais bem sucedidos ate então: o sistema MYCIN (B. G. Buchanan e E.H. Shortliffe [1985]) voltado para consulta, diagnóstico e prescrição de doencas infecciosas que apresentam bactérias ou sinais de inflamação e o sistema R1 (J. McDermott [1982]) voltado para o projeto de configurações de sistemas de computadores VAX.

Entretanto, nas tarefas de gerenciamento e controle, envolvendo a projeção, interpretação,monitoração e reelaboração de planos de ações, surgem problemas característicos dos sistemas em tempo real, delineando um conjunto de questões abordadas no trabalho de Laffey e outros:

- não-monotonicidade do conhecimento provocada pelo aspecto dinâmico do sistema (alteração sucessiva do estado provocada pela ocorrência de eventos externos);
- operação contínua;
- interface com o ambiente externo requerendo modelos de transferência e aquisição de dados;
- raciocínio temporal requerido pelas tarefas de planejamento;

- tempos de resposta rígidos;
- integração com "software" convencional e
- processamento aproximado e flexível, segundo trabalho de V.R. Lesser e outros [1988].

Embora todas estas questões sejam de fundamental importância para o perfeito funcionamento de um sistema baseado em conhecimento em tempo real , daremos ênfase, em nossa implementação, a questão citada inicialmente, relativa a garantia de tempos de resposta. Esta questão é crucial nos sistemas de produção que realizam o ciclo de reconhecimento-ação sobre um conjunto de regras, provocando um encadeamento para frente.

A análise de C. O'Reilly e A. Cromarty [1985] demonstra que os sistemas que realizam este procedimento, possuem complexidade exponencial, ou seja, o número de regras que podem ser selecionadas cresce exponencialmente com a profundidade da arvore de inferência. O mesmo resultado se aplica aos sistemas de produção com encadeamento para trás, independente da estrategia de busca ser em profundidade ou em largura.

Outro aspecto muito importante, relativo a capacidade e fexibilidade de processamento, que afeta profundamente as questões citadas, é abordado no trabalho de Esko Nuutila e outros [1987]. Neste trabalho, os autores discutem a utilização do algoritmo RETE desenvolvido por C.L. Forgy e J. McDermott [1976] e incorporado as linguagens da família OPS (OPS2 (1977), OPS4(1979), OPS83(1984) e OPS5(1981)) desenvolvidas na Universidade de Carnegie-Mellon, na programação de sistemas de produção com aplicações em tempo real.

Segundo os autores, o algoritmo RETE associado as liguagens da familia OPS não são muito favoráveis para o desenvolvimento destes sistemas. Este ponto de vista também é compartilhado pelo trabalho de K.J. Healey[1987], cuja análise demonstra que o algoritmo RETE pode ter uma performance imprevisível, não sendo possível, a priori, estabelecer um limite superior para um tempo de resposta.

No mesmo trabalho de Esko Nuutila e outros, é apresentado uma linguagem denominada XC, extensão da linguagem C++ (B. Stroustrup [1986]) , que extende à programação baseada em regras alguns princípios

da programação orientada para objetos incuindo, principalmente, a filosofia de tipos abstratos de dados (J. Guttag [1977]) e a decomposição modular de programas (D.L. Parnas [1972]).

Seguindo este raciocínio, desenvolvemos uma implementação para um sistema baseado em conhecimento que combina a utilização de sistemas baseados em regras e a programação de bases de conhecimento com a utilização de "frames". A arquitetura básica de nosso sistema é largamente influenciada pela capacidade de modularidade, abstração de dados, compilação independente e processamento concorrente da linguagem de programação Modula-2 (N. Wirth [1982]). Segundo D.R. Edelson [1987] (em um estudo comparativo de linguagens Modula-2, C++ e SMALLTALK-80, quanto a capacidade de prover mecanismos característicos da programação orientada para objetos como abstração, encapsulamento, proteção, herança e modularidade) a linguagem Modula-2, embora não seja , propriamente, uma linguagem orientada para objetos, apresenta muitos aspectos positivos, principalmente, quando voltada para a programação de grandes sistemas.

Em nossa implementação, procuramos reduzir a complexidade do processo de interpretação de regras através do uso de módulos hierárquicos de regras e algumas estratégias básicas de controle. Quanto a implementação de "frames", propusemos uma construção simples que explora a complexidade linear para operações de acesso, não permitindo, entretanto, a existência de exceções (a herança provoca a superposição ou projeção de valores), atendendo ao princípio de especialização de classes segundo G. Brewka [1987].

4.4.2 - Principais Idéias

- Algoritmo RETE:

Inicialmente, vamos comentar alguns aspectos do algoritmo RETE associado as linguagens da família OPS (OPS5 e OPS83) quando voltado para a programação de sistemas em Inteligência Artificial . Segundo W.B. Rauch-Hindin [1986] estes sistemas devem explorar o uso de múltiplos paradígmas de programação em Inteligência Artificial, permitindo a utilização conjunta de programação orientada para objetos,

programação de sistemas de produção , programação de bases de conhecimento e integração com programas convencionais.

O algoritmo RETE descrito em (C. L. Forgy [1982]) é considerado, atualmente, o melhor algoritmo para determinar o conjunto de regras aplicáveis em um sistema de produção . Segundo Nuutila e outros, a eficiência deste algoritmo é centrada em duas características assumidas para o sistema de produção:

. similaridade estrutural do lado esquerdo (antecedente) das regras e

. redundância temporal na memória de trabalho. Ou seja, somente um pequeno número de elementos da memória de trabalho pode alterar a cada ciclo.

A memória de trabalho é composta de objetos contendo um conjunto de pares atributos-valores. Cada objeto associado a uma combinação legal de par atributo-valor é denominado um elemento da memória de trabalho. Por outro lado, o lado esquerdo das regras é formado por um conjunto de padrões que descrevem elementos da memória de trabalho.

Desta forma,um interpretador de regras verifica iterativamente o casamento de padrões ou seja, se os padrões das regras descrevem elementos da memória de trabalho, determinando, assim, o conjunto de regras aplicáveis.

- Iterações sobre a Memória de Trabalho e Conjunto de Regras:

Numa visão simplista do processo de casamento de padrões, seria necessário a comparação de cada padrão com todos possíveis elementos da memória de trabalho provocando uma explosão combinatória. Entretanto, o algoritmo RETE evita este excesso de comparações, determinando para cada padrão uma lista de possíveis elementos . Estas listas são atualizadas a cada alteração da memória de trabalho, ou seja:

. se um elemento entra na memória de trabalho, o interpretador determina todos os padrões que descrevem este elemento, atualizando as respectivas listas e

. se um elemento sai da memória de trabalho, o interpretador retira o mesmo da lista de todos os padrões associados.

Desta forma, a comparação de um padrão é realizada, somente,para

um conjunto selecionado de elementos da memória de trabalho.

Para evitar a iteração sobre o conjunto de regras, o algoritmo utiliza uma rede de discriminação que indexa os casamentos parciais de padrões de todas as regras pelos respectivos elementos da memória de trabalho. Somente quando o casamento total é realizado (casamento de todos os padrões da regra) a mesma é guardada em um conjunto de conflitos. Portanto, cada alteração da memória de trabalho deve ser propagada pela rede, permitindo, desta forma, a retenção das informações de casamentos parciais de padrões e evitando a interpretação explícita a cada iteração dos componentes do lado esquerdo das regras.

- Aspectos Negativos:

1 - As modificações dos elementos da memória de trabalho devem ser propagadas através da rede de discriminação. Desta forma, como o estado da estrutura de dados é modificado a cada ciclo, durante o período de atualização, para garantir a consistência da informação, não devemos provocar interrupções no programa, o que é indesejável em aplicações em tempo real. Portanto, o acesso direto a informação por outros componentes do sistema de programação, fica seriamente afetado;

2 - O algoritmo RETE utiliza somente dados da memória de trabalho propagados pela rede. Desta forma , a representação de dados fica restrita as informações na forma objeto-atributo-valor ou elementos atualizados da memória de trabalho, tornando-se uma representação indesejável para sistemas de programação em Inteligência Artificial.

Uma proposta para a amenização destes aspectos negativos seria a utilização de uma representação múltipla de dados, ver como exemplo o sistema de produção desenvolvido por G.Bruno, A. Elia e P.Laface [1986] para a programação de tarefas em um sistema flexível de manufaturados, utilizando a linguagem OPS5 associada a um conjunto de funções externas implementadas em FORTRAN 77. Entretanto, este processo iria requerer a conversão entre diferentes tipos de representação e acarretaria o problema de inconsistência e redundância de informações.

- Representação do Conhecimento:

Na nossa implementação,procuramos desenvolver um formalismo para a representação de dados compatível com a parte convencional (procedimental) e com a parte voltada para a programação de sistemas de produção e bases de conhecimento. Para tanto, desenvolvemos uma linguagem para a representação de "frames" e elementos da memória de trabalho de tal forma que:

1 - As informações relativas aos "slots" das "frames" são utilizadas no lado esquerdo das regras comparadas a constantes ,pertencentes a conjuntos de valores legais, ou comparadas entre si, ocorrendo, neste caso, compatibilidade de tipos;

2 - As informações relativas aos "slots" das "frames" são utilizadas pela parte procedimental do programa;

3 - É possível a atualização de "frames" em tempo real observando-se a parte crítica do sistema de produção e

4 - Os elementos da memória de trabalho constituem informações dinâmicas, na forma de objetos, com tempo de vida condicionado a realização do ciclo de interpretação associado;

A implementação de nossa linguagem foi possível através da utilização do conceito de tipos abstratos de dados associado as idéia de abstração de classes desenvolvida na linguagem de programação SIMULA 67 (G. Birtwistle e outros [1973]) e de extensão de tipos (N. Wirth [1988]), permitindo a introdução dos paradígmas fundamentais da programação orientada para objetos (B. Liskov [1987]), entre os quais: encapsulamento, herança com hierarquia de tipos abstratos, instanciação com criação dinâmica de objetos, especificação de interface, implementação de métodos e proteção de informações.

Estudamos, também, o uso de um interpretador que não guarda informações parciais sobre o casamento de padrões. Entretanto, a iteração sobre os elementos da memória de trabalho para a verificação destes casamentos não se faz necessária, já que o antecedente das regras é verificado, instantaneamente, através da comparação lógica de valores provenientes de métodos e valores ativos de "frames",e dos atributos dos objetos da memória de trabalho. Desta forma, o sistema de produção incorpora os mecanismos básicos de inferência e capacidade para modelagem de objetos das "frames", melhorando, segundo R. Fikes e T.

Kehler [1985], o poder de expressão na descrição de objetos, principalmente, no que se refere a representação das relações estáticas entre os mesmos.

Quanto a redução de iterações sobre o conjunto de regras, ao invés de construirmos um esquema de indíces, determinamos uma abstração hierárquica de módulos de regras com respectivas meta-regras, que diminuem, enormemente, a complexidade do interpretador. Desta forma, ao executarmos uma meta-regra, acionamos a interpretação de um módulo de regras com um sub-objetivo específico, equivalente ao processo automático de avaliação de sub-objetivos, determinando um sub-espaço do problema (A. Newell [1980]) que seleciona o conhecimento necessário a sua solução.

Segundo R.E. Korf [1985], a estratégia citada é válida nos problemas ou sistemas de produção, onde é possível decompormos uma série de sub-objetivos ordenadamente, de tal forma que cada sub-objetivo seja dependente apenas dos sub-objetivos precedentes. A restrição proposta por R.E. Korf, que apresenta em seu trabalho uma técnica de aprendizado baseada na utilização de macro-operadores, é plenamente atendida por nossa implementação, que estabelece uma forma de decomposição hierárquica para os sub-objetivos, admitindo ainda, segundo o trabalho de E. D. Sacerdoti [1984], a definição de uma hierarquia de sub-espaços através do uso de um parâmetro de criticalidade nas premissas ou componentes dos antecedentes das regras.

Recentemente, Rosembloon e Forgy desenvolveram uma nova versão de linguagem da família OPS conhecida como OPS6. Embora ainda não implementada (não temos maiores informações), esta versão inclui a possibilidade de listas e conjuntos serem utilizados como estruturas de dados básicas da memória de trabalho. Esta nova versão permitirá, também, a existência de múltiplas memórias de trabalho e produções.

- Decomposição e Modularidade:

A decomposição de programas em módulos e tipo abstrato de dados é, reconhecidamente, uma das técnicas mais eficientes de programação, permitindo a redução da complexidade de sistemas através do uso de mecanismos de abstração segundo D.L. Parnas [1972] e B. Liskov, J.

Guttag [1986].

Nos sistemas de produção, esta técnica não é atendida, derivando um complexo conjunto linear de regras associado a uma base global de dados (memória de trabalho).

Em nossa implementação, através da construção de módulos hierárquicos de regras associados a utilização de tipos abstratos de dados, apresentamos uma filosofia de programação orientada para objetos. Para tanto, utilizamos as facilidades para a decomposição de programas em módulos proporcionadas pela linguagem Modula-2. Comparada a linguagem XC (Esko Nuutila e outros [1987]), a linguagem Modula-2 apresenta as seguintes vantagens:

- sintaxe e semântica das construções mais bem definidas, proporcionando maior simplicidade e confiabilidade aos programas;
- compilação independente incluindo a análise de compatibilidade de interfaces;
- somente uma fase de compilação;
- primitivas para programação concorrente atraves do uso de co-rotinas;
- orientada para construção de módulos (lista de recursos exportáveis/importáveis e definição/implementação separada de modulos);
- existência de um bom ambiente de programação (Logitech Modula-2).

Utilizando a linguagem XC, necessitamos converter, inicialmente, o programa para a linguagem C++. Em seguida, o mesmo programa deve ser convertido para a linguagem C, possibilitando, finalmente, a tradução para o código de máquina. Este processo de compilação apresenta várias deficiências, entre as quais:

- inexistência de uma especificação formal para a linguagem base C++;
- inexistência de "debugger" para a análise dos programas em XC e C++;
- lentidão do processo de compilação, e
- geração excessiva de código, um programa com 30 Kbytes de código XC gera, aproximadamente, 400 Kbytes de código na linguagem C.

4.4.3 - Desenvolvimento e Implementação

- Estrutura de Representação do Conhecimento:

O sistema de representação do conhecimento que utiliza "frames" (M. Minsky [1975]) é uma forma de representação baseada em objetos que inclui dois processos fundamentais de organização do conhecimento, os quais definem a interrelação entre objetos : especialização e composição.

A linguagem desenvolvida para a representação de "frames" , conforme já mencionamos, é baseada na idéia de abstração de classes, permitindo uma taxonomia ou hierarquia de objetos incluindo meta-classes, classes e instâncias terminais. Seguindo a nomenclatura de G. Brewka, consideramos meta-classes classes cujas instâncias (variáveis de classe) são classes. Desta forma, existem em nossa linguagem apenas dois tipos de objetos: classes e instâncias terminais, já que instâncias terminais não representam abstrações. Para representarmos esta taxonomia utilizamos dois construtores fundamentais que vão definir o relacionamento de classes: herança ou princípio de especialização e instanciação, que permite a criação dinâmica de objetos a partir de classes.

Uma classe descreve a estrutura de um conjunto de objetos, através dos seguintes tipos básicos de "frames":

```
DefClasse  < nome >:
SlotSuper : ponteiro para DefClasses;
Slots:
  Slot1 : tipo1
  Slot2 : tipo2
 ...
  Slotm : tipom
FimClasse.
```

Esta "frame" define uma classe que herda características de outras classes e possuí um conjunto de slots que são atributos próprios. Caso não haja referência para alguma classe (ponteiro para NIL) a "frame" representa a definição de uma super-classe ou seja, uma classe situada no nível mais alto da hierarquia.

```
InstClasse < NomeInstância >:
SlotSuper <-  NIL;
SlotNome : < nome DefClasse >;
Slots:
  Slot1 <- valor1
  Slot2 <- valor2
 ...
  Slotm <- valorm
FimClasse.
```

Esta "frame" representa uma instância de classe que não possui herança, podendo ser uma classe se for referenciada por outro objeto ou instância de nova classe. Neste caso a classe que define é uma meta-classe.

Como permitimos instâncias de classes serem classes podemos ter ,também, a seguinte estrutura:

```
InstClasse < Nome Instância >;
SlotNome: < nome DefClasse >;
SlotsHerdados <-  ponteiro para
  InstClasse < NomeInstância > com :
    Slot1 <- valor1
    Slot2 <- valor2
   ...
    Slotm <- valorm;
Slots:
  Slotm+1 <- valorm+1
  Slotm+2 <- valorm+2
 ...
  Slotm+n <- valorm+n
FimClasse.
```

Esta "frame" representa um objeto que herda todos os valores específicos de uma instância de classe e possuí, também , um conjunto de valores específicos que o especializam a partir de uma determinada classe. Caso o objeto acima não seja referenciado por nenhuma instância de classe, o mesmo é considerado instância terminal.

É importante ressaltarmos que na nossa linguagem a distinção entre metaclasses, classes e instâncias terminais depende exclusivamente

da ocorrência ou não de herança.

Analisando as informações providas pelos "slots" podemos ter:

. "slots" que descrevem atributos gerais de uma classe de objetos. Neste caso, as informações determinam um tipo associado que pode impor restrições de cardinalidade e domínio (valores legais) aos possíveis valores atribuídos. A representação do tipo deve ser compatível com qualquer tipo de dado ordinal (conjunto, enumerado, sub-alcançe, lógico, caractere e inteiro), estruturado ("arrays" e registros) e real;

. "slots" que descrevem atributos específicos (valores) de objetos (instâncias de classes). Neste caso, as informações determinam um valor compatível com as restrições de tipo impostas pelo construtor da classe (DefClasse) . Para os valores do tipo ordinal, real e algum componente de tipo estruturado cujo tipo base, também, seja ordinal ou real, o acesso é provido por métodos ou funções que retornam um valor do tipo cardinal através do uso das funções pré-definidas ORD e TRUNC.

Neste caso, temos dois tipos de valores: valores próprios, característicos da classe e valores herdados característicos de uma classe de objetos. Caso o valor seja do tipo estruturado o acesso é provido por procedimentos com o uso de parâmetros passados por referência;

. "slots" que expressam o relacionamento entre "frames" relativo a pertinência (condição de membro ou instância de uma classe), e

. "slots" que expressam o relacionamento de herança, ou seja, a capacidade de herdar de uma classe suas propriedades, métodos e relacionamentos.

Podemos, também, associar aos "slots" informações na forma procedimental ou, segundo Fikes e Kehler, valores ativos. Estes valores ativos são retornados por procedimentos ou conjunto de regras de produção acionados, automaticamente, quando acessamos os "slots". Na nossa implementação, foi possível associar procedimentos e conjunto de regras aos "slots" através do uso de variáveis do tipo PROCEDURE, permitido pela linguagem Modula-2, e pela possibilidade de definirmos abstrações funcionais em módulos de regras .

Outra questão importante, que devemos analisar na representação de "frames", diz respeito ao tipo de especialização, quanto ao fato de ser ou não, mutuamente exclusiva (Charles P. Wagner [1988]). Em nossa implementação permitimos a instanciação discriminada de objetos pelo uso de um "slot" que determina a variante escolhida, ou seja:

```
Slot1 : tipo1;
...
Slotp-1 : tipop-1;
Caso Slotp : tipop
opção1 : Slotp+1 : tipop+1 ... Slotp+n : tipop+n;
...
opçãom : Slotp+1 : tipoP+1 ... Slotp+n : tipoP+n
FimCaso.
```

Entretanto, esta forma de representação não permite a ocorrência de exceções no mecanismo de herança, mas somente uma instanciação discriminada, mutuamente exclusiva, em relação aos "slots" $1,\ldots,p-1$, admitindo a ocorrência de superposição em relação aos "slots" $p,\ldots,p+n$.

- Implementação de "Frames":

A implementação clássica de um tipo abstrato de dado requer uma representação apropriada e um conjunto de operações associados a uma estrutura de dados, bem como o projeto de uma interface que determinará a funcionalidade da estrutura em relação ao meio externo.

Tradicionalmente, este processo é concebido em módulos, com interface e implementação separados, permitindo desta forma a proteção e controle de visibilidade de informações e procedimentos.

O processo básico de construção de tipo abstrato de dados foi utilizado em nossa linguagem para a implementação de "frames". Associada a representação das "frames", desenvolvemos um conjunto de métodos que permitem o acesso instantâneo aos valores dos "slots" e procedimentos para a criação, destruição e manipulação das mesmas.

Utilizando a sintaxe da linguagem de programação Modula-2 vamos descrever a implementação de algumas representações de frames sugeridas

anteriormente:

```
DEFINITION MODULE frame1;
EXPORT QUALIFIED Ptframe1, Criaframe1;
TYPE     Ptframe1 = POINTER TO RECORD
                        SlotSuper: POINTER TO DefClasse;
                        SlotNome: ARRAY OF CHAR;
                        Slot : tipo
                            1       1
                        ...
                        Slot : tipo
                            m       m
                    END;
PROCEDURE Criaframe1( lista de valores dos slots ): Ptframe1;
END frame1.


IMPLEMENTATION MODULE frame1;
PROCEDURE Criaframe1( lista de valores dos slots ): Ptframe1;
VAR InstFrame: Ptframe1;
BEGIN
  NEW(InstFrame);
  WITH InstFrame^ DO
    SlotSuper <- NIL
    SlotNome <-  < nome DefClasse >
    Slot     <-  valor
        1              1
    ...
    Slot     <-  valor
        m              m
  END;
  RETURN InstFrame
END Criaframe1
END frame1.

DEFINITION MODULE frame2;
FROM frame1 IMPORT Ptframe1 ;
EXPORT QUALIFIED Ptframe2, Criaframe2, Sloti  , ... , Slotm+n;
TYPE  Ptframe2;   (*  tipo opaco *)
PROCEDURE Criaframe2 ( lista de valores dos slots ): Ptframe2;

(* especificação dos métodos *)
PROCEDURE Sloti ( Pt : Ptframe2 ) : tipo ;
                                        1
...
PROCEDURE Slotm+n   (Pt : Ptframe2 ) : tipo   ;
                                          m+n
END frame2.

IMPLEMENTATION MODULE frame2;
FROM frame1 IMPORT Ptframe1;
TYPE Ptframe2 = POINTER TO RECORD
                    SlotSuper: POINTER TO DefClasse;
                    SlotNome: ARRAY OF CHAR;
```

```
                    Slot : tipo
                       m+1      m+1
                    ...
                    Slot : tipo
                        m+n     m+n
                END;

PROCEDURE Criaframe2 ( lista de valores dos slots ): Ptframe2;
VAR InstFrame: Ptframe1;
BEGIN
  NEW(InstFrame);
  WITH InstFrame^ DO
    SlotSuper <- Instframe1;
    SlotNome <-  < nome DefClasse >;
    Slot     <-  valor
        m+1            m+1
    ...
    Slot     <-  valor
        m+n            m+n
  END;
  RETURN InstFrame
END Criaframe2;

(* métodos *)
PROCEDURE Slot1 ( Pt : Ptframe2 ) : tipo  ;
                                          1
BEGIN
  RETURN Pt^.SlotSuper^.Slot
                           1
END Slot1;
...
PROCEDURE Slotm+n ( Pt : Ptframe2 ) : tipo   ;
                                        m+n
BEGIN
  RETURN Pt^.Slot
               m+n
END Slotm+n;
END frame2.


DEFINITION MODULE frame3;
FROM frame1 IMPORT Criaframe1, Ptframe1;
FROM frame2 IMPORT Criaframe2, Ptframe2;
EXPORT QUALIFIED Instframe1, Instframe2, InstFrame;
VAR Instframe1: Ptframe1;
    Instframe2: Ptframe2;
PROCEDURE InstFrame()
END frame3.

IMPLEMENTATION MODULE frame3;
FROM frame1 IMPORT Criaframe1, Ptframe1;
FROM frame2 IMPORT Criaframe2, Ptframe2;
PROCEDURE InstFrame();
BEGIN
  Readframe1 ( lista de valores dos slots ) ;
```

```
        Instframe1 <- Criaframe1 ( lista de valores dos slots );
        Readframe2 ( lista de valores dos slots );
        Instframe2 <- Criaframe2 ( Instframe1,
                                   lista de valores dos slots );
    END InstFrame
    END frame3.
```

A implementação do mecanismo de herança foi baseada na idéia de extensão de tipos (N. Wirth [1988]), ou seja, na construção de novos tipos a partir de tipos existentes. Portanto:

Se T = RECORD x,y END e T' = RECORD (T) z END ,

então dizemos que o tipo T' é uma extensão direta do tipo base T (T' -> T), sendo que o modulo de T' importa de um tipo abstrato de dados o tipo base T e os procedimentos necessários a sua manipulação.

Os valores de um tipo extendido T' superpõe os valores do tipo base T respeitando o princípio de especialização . Por transitividade, é possível definirmos uma hierarquia de tipos,semelhante a hierarquia de classes descrita anteriormente. Wirth propõe, também, uma extensão de tipos para ponteiros, ou seja:

Se P --> T (P se liga a T) e T' -> T (T é um tipo base de T'),

então podemos afirmar que P'--> T' (P' ligado a T') extende o ponteiro base P (P'-> P).

Este processo de extensão de tipos e ponteiros é sugerido por Wirth na linguagem Oberon (N. Wirth [1988]) que é considerada uma evolução da linguagem Modula-2 em relação a programação orientada para objetos.

J. Bergin e S. Greenfield [1988] discutem as facilidades suportadas pela linguagem Modula-2 para a programação orientada para objetos incluindo as questões de encapsulamento, herança e troca de mensagens. Apesar de Modula-2 não ser, propriamente, uma linguagem objeto orientada , os autores destacam a grande viabilidade de se utilizar um estilo de programação orientada para objetos, através da implementação de um mecanismo de herança que incorpora o processo de

extensão de tipos e ponteiros de N. Wirth.

Neste trabalho, os autores também destacam algums aspectos deficientes da linguagem, ressaltando o problema de criação e inicialização de novos objetos. Este problema, exaustivamente discutido na literatura de Modula-2 (R.S. Wiener [1986], J. Savit [1987] e S. Greenfield e R. Norton [1987]) , pode ser contornado se definirmos um procedimento que cria e , automaticamente, inicializa novos objetos, conforme a proposta de nossa implementação.

Outro problema, mais sério, se refere a questão da restrição de visibilidade dos componentes de uma entidade classe fora da hierarquia de heranças. Em nossa implementação, tornamos os componentes da estrutura que define classe de trens, totalmente visíveis para qualquer módulo que queira importar objetos desta classe, já que, em Modula-2 não é possível restringir os recursos exportáveis a determinados módulos.

Entretanto, o acesso aos componentes de algum objeto da classe de trens só pode ser feito intencionalmente, já que é necessário declarar nos modulos clientes a importação destes objetos. Desta forma, na nossa implementação, permitimos o acesso aos componentes da estrutura classe de trens somente nos módulos que estabelecem a hierarquia de heranças.

É importante ressaltarmos que todas as estruturas que representam instâncias terminais são implementadas como tipo opacos (visibilidade protegida) pois pertencem ao último nível da hierarquia de heranças.

- Memória de Trabalho:

Atualmente, todas as visões sobre a arquitetura do sistema de processamento de informações humano concordam com a distinção de dois componentes básicos da memória :

. um tipo de memória com quantidade limitada e altamente dinâmica e

. um tipo de memória de grande capacidade e de comportamento quase estático.

O primeiro tipo de memória, na nomenclatura dos sistemas de

produção, é denominado memória de trabalho . Geralmente, a memória de trabalho é constituída de uma coleção de elementos ou informações que representam determinados estados ou contextos no processo de interpretação de um conjunto de produções.

Para representarmos estas informações, utilizamos uma coleção de triplas, na forma objeto-atributo-valor, implementadas através de tipos abstrato de dados dinâmicos . Ou seja, a nossa memória de trabalho é constituída de um conjunto de ponteiros para uma estrutura que descreve uma classe de objetos que possuem um conjunto de atributos associados a valores legais de conjuntos enumerados.

Portanto, definimos um mecanismo de abstração e encapsulamento que comporta uma estrutura de representação, um conjunto de operações de manipulação (cria, remova, faça, atributo, modifica, etc) e conjuntos de valores legais.

Como exemplo, suponha que queiramos representar na memória de trabalho elementos que descrevem o objeto trem com atributos carga e tipo. Na sintaxe da linguagem Modula2 teremos:

```
DEFINITION MODULE MemoriaTrabalho;

EXPORT QUALIFIED Ptobjeto, cria, faca, remova, atributo,
ModificaAtributo, PrintAtributo, LegValTipo, LegValCarga;

TYPE Ptobjeto;  (* tipo opaco *)

     LegValTipo = (NaoDefinida,unitario,servico,CargaGeral);
     LegValCarga = (naodefinida,minerio,carvao,gusa,cimento);

PROCEDURE cria ( VAR p: Ptobjeto; nome: ARRAY OF CHAR );
PROCEDURE faca ( VAR p: Ptobjeto; a,c: CARDINAL);
PROCEDURE atributo (p: Ptobjeto; a: CARDINAL): CARDINAL;
PROCEDURE remova (VAR p: Ptobjeto);
PROCEDURE ModificaAtributo (VAR p: Ptobjeto; tex:ARRAY OF CHAR);
PROCEDURE PrintAtributo (p: Ptobjeto);

END MemoriaTrabalho.


IMPLEMENTATION MODULE MemoriaTrabalho;

TYPE ObjPtr = POINTER TO TipObj;

TipObj =  RECORD
   recency, cont: CARDINAL;
   atributo: ARRAY [1..3] OF CARDINAL;
```

```
      nome: ARRAY [0..60] OF CHAR;
    END;

  Ptobjeto = ObjPtr;

  (* implementação de procedimentos *)

  END MemoriaTrabalho.

  MODULE exemplo;
  FROM MemoriaTrabalho IMPORT Ptobjeto, cria, remova, faca,
  atributo, LegValTipo, LegValCarga;
  VAR trem: PtObjeto;
      CargaTrem : CARDINAL;
  BEGIN
      cria (trem,' MeuTrem ');
      (* cria objeto trem *)
      faca (trem,1,1);
      (* atribue ao tipo o valor unitario *)
      CargaTrem <- atributo(2,trem);
      (* atribue a CargaTrem o valor do atributo carga *)
      remova(trem);
      (* remove o objeto trem da memória de trabalho *)
  END exemplo.
```

Portanto, a nossa implementação permite a representação de múltiplos objetos contendo múltiplos atributos, podendo cada atributo conter somente um valor legal de um conjunto enumerado de valores legais. Esta restrição sobre o conjunto de valores legais é imposta pelo interpretador de regras. Entretanto, como qualquer tipo ordinal pode ser considerado como um conjunto enumerado de valores, a mesma não apresenta maiores problemas. Além do mais, o papel dos objetos fica restrito a representação de informações dinâmicas necessárias ao encadeamento das regras. Sendo assim, uma instância de objeto associada a um atributo e um valor legal representa um elemento da memória de trabalho.

Caso necessário, podemos permitir a implementação de objetos contendo múltiplos valores para atributos. Neste caso, uma instância de objeto pode descrever vários elementos, aumentando, desta forma, o numero de comparações no processo de reconhecimento das regras.

Associado a representação dos objetos, temos uma informação referente a ordem de alteração dos mesmos na memória de trabalho. Esta informação possibilita ao interpretador resolver o conflito de regras,

observando os componentes das mesmas e dando prioridade aquela cujos padrões se ligam a elementos da memória de trabalho que foram modificados mais recentemente.

- Regras e Modulos de Regras:

Uma regra de produção em nosso sistema e uma estrutura da forma:

Quando

condição$_1$ E E condição$_n$ E (condição$_{n+1}$ OU...OU condição$_{n+m}$)

Então

ação

Ou seja, é permitida a regra na forma normal conjuntiva com o uso de conectivos E e OU isoladamente, e o uso combinado dos conectivos observando a forma apresentada, de forma que o interpretador de regras abandone o processo de reconhecimento sempre que uma condição ou disjunção de condições for falsa.

Entretanto, no lado direito da regra (ação desejável) é permitido apenas um consequente, que provoca uma alteração em elementos da memória de trabalho. Para verificarmos melhor a representação de regras, vamos apresentar alguns exemplos, utilizados em regras específicas de nosso sistema :

```
When (sentido(trem[t1]),igual,ORD(par),1,36);
  And (sentido(trem[t2]),igual,ORD(impar),1);
Then (faca,conflito,1,ORD(cruzamento),36);

When (atributo(1,conflito),igual,ORD(cruzamento),1,1);
  And (NumLinhas(seção[PosSeção]),igual,1,1);
Then (faca,objetivo,1,PrefParar(PosSeção,t1,t2),1);
```

Na primeira regra, composta de duas condições e um consequente, podemos observar que uma condição pode ser formada por comparações lógicas (maior, menor, igual e diferente) de valores ativos relativos a instâncias indexadas de frames que representam trens. O consequente da mesma determina a atribuição do valor 'cruzamento' ao primeiro atributo do objeto conflito , determinando a criação de um elemento na memória de trabalho.

Na segunda regra, observamos que a primeira condição é referente ao padrão (o atributo$_1$ de conflito é 'cruzamento') que é verificado, também, através da comparação lógica do valor do atributo$_1$ de conflito a constante 'cruzamento' pertencente a um conjunto enumerado de valores legais. Quanto ao consequente da segunda regra, observamos a existência de uma função que representa uma abstração funcional de um módulo de regras, determinando, portanto, o papel de uma meta-regra.

Esta função retorna um valor compatível com o atributo$_1$ do objeto 'objetivo', sendo importada de um módulo de regras específico . Desta forma, podemos dizer que um módulo de regras na sintaxe de Modula-2, tem a forma:

```
DEFINITION MODULE exemplo;
EXPORT QUALIFIED subobjetivo;
PROCEDURE subobjetivo (lista de parâmetros): CARDINAL;
END exemplo.

IMPLEMENTATION MODULE exemplo;
FROM MODULE < módulo de regras > IMPORT < abstrações funcionais >;
PROCEDURE subobjetivo (lista de parâmetros ): CARDINAL;
(* declaração de variáveis *)
BEGIN
  (* inicialização de variáveis e
     de parâmetros do interpretador,
     inicialização da pilha de raciocínio e
     criação dos  objetos da memória de trabalho *)

  LOOP

      (* regras e meta-regras *)

  END;

  (* destruição dos elementos da memória de trabalho
  (* explanação de raciocínio e
     destruição da pilha  *)
  RETURN subobjetivo
END subobjetivo;
END exemplo.
```

A partir desta representação, podemos fazer as seguintes observações:

- o consequente das regras modifica somente valores da memória de trabalho (conhecimento dinâmico);

- a atualização de "frames" é realizada externamente, observando as seções críticas do interpretador;

- cada módulo de regras define seu próprio ambiente ou sub-espaço de conhecimento , possuindo um conjunto próprio de regras, elementos da memória de trabalho e um sub-objetivo específico determinando uma abstração funcional;

- é possível estabelecer uma hierarquia de módulos;

- é permitida a parametrização dos módulos de regras associando as respectivas funções, parâmetros que indexarão variáveis de "frames";

- cada regra, em resumo, é constituída de um conjunto de funções que testam a validade dos consequentes e executam a ação desejável;

- é determinado para cada modulo um número máximo de iterações permitido. Caso o número não seja suficiente, a função retorna o valor 0 (zero) correspondente a ordem da constante inicializada para o objeto 'subobjetivo'.

- Funcionamento:

Cada módulo de regras, quando compilado, produz um código executável associado a um procedimento (abstração funcional) que representa um sub-objetivo específico.

Quando o procedimento é invocado por alguma meta-regra ou outra parte do programa ("slot" de "frame", algoritmo de busca heurística, etc.) o mesmo determina a criação de ambiente de conhecimento e inicia o "loop" de reconhecimento-ação. O "loop" prossegue até a execução da função "RETURN". Neste instante, temos duas possibilidades:

1- foi determinado um valor legal para o subobjetivo, ou

2- não foi encontrada uma regra aplicável e excedeu-se o numero de repetições permitidas (normalmente, consideramos o número máximo igual ao número de regras).

A função RETURN provoca a transferência de controle para o ponto do programa que determinou a abstração funcional (algum módulo de regra situado em um nível hierárquico superior ou outra parte do programa).

Tradicionalmente, os interpretadores de produções, realizam em primeiro lugar, o processo de interpretação de regras, guardando as regras plenamente satisfeitas em um conjunto de conflitos. Em seguida, o interpretador seleciona uma regra do conjunto de conflitos, segundo uma estratégia desejável, e provoca a ação associada ao consequente da mesma.

Na nossa implementação, como iteramos sobre o conjunto de regras , é possível guardarmos, seletivamente, a melhor instância de regra ou consequente aplicável, observando os seguintes critérios de preferência (D. Rosenthal [1985]), já apresentados na seção 4.3.2:

1 - preferência para a regra ou meta-regra não executada recentemente ("refraction");

2 - preferência para meta-regra quando em conflito com regras (controle procedimental) ;

3 - preferência para a regra cujo componente associado a um objeto da memória de trabalho tenha sido alterado mais recentemente ("recency").

Existe, também, uma quarto critério (ver seção 4.3.2), não implementado, que determina a preferência na regra que contenha maior número de componentes no lado esquerdo (especificidade).

Em nossa implementação possibilitamos, também, a exemplo de J. Robertson [1985] , o uso de uma estratégia simples que seleciona a primeira regra aplicável para execução, obedecendo aos critérios determinados anteriormente, dando maior velocidade de execução ao interpretador.

Finalmente, queremos relembrar a possibilidade de implementarmos em um módulo de regras, hierarquias de espaços, através do uso de parâmetros de criticalidade nos componentes das regras. Portanto, temos durante o processo de interpretação de um módulo, um nível crítico que é comparado aos parâmetros de criticalidade das regras e decrescido quando não houver mais regras aplicáveis neste nível , até um valor correspondente a um nível mínimo . Neste caso, todo o espaço do problema ou componentes das regras serão considerados.

- Explanação do Raciocínio:

Segundo A. Newell e H. Simon [1972], os sistemas de produção e, consequentemente, os sistemas baseados em conhecimento, apresentam uma série de vantagens sobre os sistemas de programação convencional, sendo mais indicados para a modelagem do processo humano de raciocínio. Entre as principais características que proporcionam estas vantagens, podemos destacar :

. modularização do conhecimento em unidades homogêneas (regras);

. incorporação do conhecimento de especialistas (heurísticas);

. capacidade de incrementar o conhecimento (independência das regras);

. modelagem da memória (memória de curto termo e memória de longo termo);

. conduta objetiva voltada para a solução de problemas e

. mecanismos de explanação sobre conclusões e linhas de raciocínio.

O objetivo dos mecanismos de explanação é possibilitar a cada consulta, sabermos Por Que? e Como? foi determinada alguma das conclusões associadas as respectivas linhas de raciocínio.

No nosso sistema, afim de possibilitarmos a realização de explanação, desenvolvemos uma estrutura recursiva, que armazena a forma textual das regras utilizando como chave seus respectivos números. Ou seja, armazenamos uma coleção de ponteiros para objetos do tipo regra. Na sintaxe de Modula-2 temos:

```
TYPE   EPtr = POINTER TO EObj;
       EObj = RECORD
                nome: string;
                next: EPtr
              END;
       OuPtr = POINTER TO OuObj;
       OuObj = RECORD
                nome: string;
                next: OuPtr
               END;
       ObjRegra = RECORD
                    NomeQuando: string;
                    NomeE: EPtr;
                    NomeOu: OuPtr;
                    NomeEntão: string;
                    NomeModulo: TipNome;
                  END;
        RegraPtr = POINTER TO ObjRegra;
```

Desta forma, ao inicializarmos o sistema, criamos uma lista de regras, que são lidas sequencialmente de um arquivo.

Durante o processo de interpretação, é colocado na pilha de raciocínio o numero indexador das regras que são ativadas. Portanto, é possível, através do uso do consequente e antecedente das regras, explicar Por Que ? e Como ? chegamos a determinadas conclusões, imprimindo textualmente estas informações que são obtidas através da realização de uma pesquisa sequencial na lista de regras tendo como chaves os respectivos valores da pilha .

Ou seja, ao perguntarmos Por Que ? o sistema chegou a determinada conclusão, recebemos a informação dos consequentes das regras. Ao perguntamos Como ? o sistema chegou a determinada conclusão (através da ação dos consequentes) , recebemos a informação dos antecedentes das regras.

É permitido, também, ao usuário do sistema :

. visualizar textualmente qualquer regra ou meta-regra e seu respectivo módulo;

. consultar e modificar os "slots" das "frames" e respectivos valores;

. consultar e modificar os atributos e valores dos objetos durante o processo de interpretação.

Devido ao fato dos módulos de regras, bem como, das estruturas "frames" serem compilados, sacrificamos, em nossa implementação, a facilidade de modificação e extensão do conhecimento, em função de maior eficiência computacional. Entretanto, se verificarmos as facilidades providas pela linguagem Modula-2 para a compilação independente e o alto grau de modularização do conhecimento, veremos que este sacrifício é extremamente vantajoso, principalmente, tratando-se do desenvolvimento de um sistema específico que exige poucas mudanças.

Se tivermos que alterar uma regra específica, ou adicionarmos uma nova regra ao sistema, teremos que recompilar novamente somente o módulo associado a respectiva regra. É necessário, também, a fim de manter a consistência dos mecanismos de explanação, atualizarmos

adequadamente o arquivo de regras na forma textual.

4.4.4 - Programação com Múltiplos Paradígmas

- Ambiente de Pogramação:

D.G. Bobrow [1985] questiona o fato de que um simples paradígma ou estilo de programação não e suficiente para a programação de sistemas de conhecimento. Ao contrário, é cada vez mais necessário a integração de diferentes paradígmas de programação dentro de um ambiente amigável e flexível ao usuário. Entre as facilidades que este ambiente deve proporcionar, Bobrow destaca:

- "debbuging" e "run-time debugging";
- capacidade de comunicação interativa com o usuário;
- interface amigável;
- capacidade de desenvolvimento incremental e
- facilidades para monitoração e análise.

A existência destas facilidades é de fundamental importância no desenvolvimento de bases de conhecimento, onde, não sabemos, a priori, os requerimentos necessários a sua elaboração, ou temos o conhecimento na forma incompleta. Neste sentido, o desenvolvimento de programas envolve um contínuo processo de refinamento e realimentação . Conforme enfatizado por H. Simon [1981], a Ciência de Computação e o desenvolvimento evolutivo de programas de computador constituem atividades, essencialmente, empíricas, fruto de experimentos e acertos. B. Sheil [1984] refere a este tipo de programação como um processo de programação exploratório.

- Programação com Lógica:

Em seguida, Bobrow faz uma análise dos principais paradígmas de programação , incluindo a programação orientada para objetos, orientada para processos, programação com lógica e programação baseada em regras, destacando suas qualidades e abordando algumas deficiências.

A respeito das vantagens e deficiências do paradígma de programação com lógica (J. Hogger [1984]) e, em particular, da

linguagem Prolog, quando voltada para a programação de bases de conhecimento, achamos por bem destacar alguns pontos descritos no trabalho de P.A. Subrahmanyam [1985]:

. Aspectos positivos:

. suporta controle não-determinístico do conhecimento, através de um esquema de encadeamento de regras para trás associado a estratégia de busca em profundidade com realização de "backtracking";

. utilização do método de resolução como forma de solução de problemas através da refutação de hipóteses (satisfabilidade de fórmulas);

. uso de unificação sintática ou casamento de padrões , suportando um estilo de programação baseado em cláusulas (Horn-Clause) simples e elegante, que permite a programação de sistemas baseados em regras;

. representação do conhecimento factual e condiconal na forma declarativa através de setenças incluindo assertivas e implicações contendo termos e predicados.

. Aspectos negativos:

. falta de flexibilidade e controle de conhecimento no processo de busca durante a execução do programa, gerando um espaço de computação excessivo. Algumas implementações de Prolog utilizam o predicado de controle "cut" como forma de reduzir o processo de busca. Entretanto, o uso deste predicado, considerado por J.A. Robinson [1983] como o "go to" do Prolog, afeta a semântica declarativa do programa;

. inadequado para problemas que requerem o encadeamento de regras para frente;

. falta de modularidade afetando o desenvolvimento de grandes programas;

. existência de uma única base global de dados;

. falta de informação a respeito da corretude do algorítmo, podendo levar a existência de um "loop" sem derivar uma resposta lógica do problema (cláusula vazia). Embora esta questão seja proporcionada pela limitação de decidibilidade da lógica de predicados e computabilidade de funções (Z. Manna [1974]); na linguagem Prolog, este problema é mais grave, pois quando o programa não termina com soluções satisfatórias (totalmente correto) não podemos argumentar

sobre a corretude parcial do mesmo (especificação consistente);

. programação baseada em cláusulas é inadequada para certos tipos de problemas.

- Programa SIGT:

O programa SIGT foi desenvolvido de forma modular, para processadores da linha INTEL 8088, 80286 e 80386 e sistema operacional DOS, segundo vários paradígmas de programação, incluindo a programação orientada para objetos, programação baseada em regras e programação procedimental.

O programa contém, em sua versão atual (que realiza as funções de planejamento, monitoração e simulação), aproximadamente 12000 linhas (300 KBytes divididos em 63 módulos básicos) de código Modula-2 (Logitech Modula-2/86 versões 2.0 e 3.0).

Modula-2 é uma poderosa, simples e eficiente linguagem de programação, desenvolvida por Niklaus Wirth [1982], permitindo a realização de compilação separada, desenvolvimento de tipos abstratos de dados, programação concorrente e programação em baixo nível (G. A. Ford e R.S. Wiener [1985]). G. Barney [1986] considera Modula-2 a melhor linguagem para o desenvolvimento de sistemas de programação em grande porte em tempo real ,quanto aos critérios de portabilidade, eficiência, redigibilidade, confiabilidade, flexibilidade e simplicidade, quando comparada as linguagens Coral 66, RTL/2, ADA e Pascal.

Atualmente, existem um grande número de ferramentas (sistemas híbridos de programação) voltados para a construção de programas em Inteligência Artificial que utilizam múltiplos paradígmas. Dentre as ferramentas mais importantes devemos destacar três:

1 - KEE ("Knowledge Engineering Environment") (T.P.Kehler e G.D. Clemenson [1984]). KEE é um exemplo de programação orientada para objetos, tendo como base a linguagem LISP, incluindo a utilização combinada de regras e "frames". Foi desenvolvido pela Intellicorp, que é considerada a empresa pioneira na produção de sistemas de programação inteligentes;

2 - LOOPS (M. Stefik e outros [1983]). Desenvolvida pela XEROX PARC ("Paolo Alto Research Center") e baseada no uso da linguagem INTERLISP. A ferramenta LOOPS é, também, um exemplo de programação orientada para objetos que suporta programação procedimental e desenvolvimento de regras;

3 - SRL ("Schema Representation Language") ou Knowledge Craft (M.S. Fox e J. McDermott [1986]). Desenvolvida pelo grupo de Inteligência Artificial da Universidade de Carneggie-Mellon, e implementada em Common LISP. SRL incorpora uma linguagem para construção de "frames" e uma linguagem , SRL-OPS, para o desenvolvimento de sistemas de produção.

O aspecto negativo destas ferramentas, é que todas são implementadas em dialetos da linguagem LISP, estando disponíveis apenas em máquinas LISP (Xerox 1100, Symbolics 3600, TI Explorers, estações Appolo e Sun, UNIX VAX-LISP), sendo voltadas, essencialmente, para a programação de bases de conhecimento, não possibilitando a integração de métodos de raciocínio e busca, existente, por exemplo, na arquitetura SOAR (J.E. Laird,A. Newell e P.S. Rosenbloom [1987]).

Entretanto, o programa SOAR, 255 KBytes de código LISP (CommonLISP, ZETALISP,INTERLISP,FRANZLISP), possui o mesmo aspecto restritivo das ferramentas anteriores em relação a implementação em máquinas LISP.

Convém ressaltarmos, que, quanto ao desenvolvimento de sistemas especialistas, existe o desenvolvimento de um sistema aplicado a guerra eletrônica "SEGE" (F.C.C. Dargam, E.P.L. Passos e F.R. Pantoja [1988]) que utiliza uma implementação da linguagem OPS5 (F.R.D. Velasco [1987]) para ambiente de micro-computadores compatíveis com o IBM-PC/AT.

- Reflexões:

Neste capítulo, apresentamos uma proposta de arquitetura para o desenvolvimento de sistemas baseados em conhecimento que integram sistemas inteligentes voltados para funções de gerenciamento e controle.

A implementação destes sistemas é viável em qualquer linguagem de alto nível, que suporte a programação de tipos abstrato de dados,

como ADA, CLU, SIMULA-67 e C++. Entre as principais vantagens de nossa arquitetura destacamos:

- utilização combinada de "frames " e produções;
- utilização de memória dinâmica;
- concepção hierárquica de módulos de regras e uso de meta-regras;
- ausência de explosão combinatória no processo de comparação de padrões com elementos da memória de trabalho;
- ausência de inconsistência e redundância de dados acarretados pela existência de múltiplas representações;
- possibilidade de programação em múltiplos paradígmas;
- possibilidade de programação em tempo real;
- desenvolvimento em linguagem de alto nível Modula-2.

Neste sentido, queremos comentar alguns aspectos do trabalho de H. Simon [1986] que possibilitaram algumas reflexões a respeito de nosso sistema. Segundo Hebert Simon, a Inteligência Artificial surgiu da necessidade de gerar, através do uso de programas de computador, boas soluções para problemas mal estruturados, ou seja, problemas cuja estrutura não foi suficientemente formalizada de modo a permitir, através do uso de técnicas tradicionais de modelagem e análise, a obtenção de soluções ótimas em tempos de computação razoáveis. Para tanto, estes programas, incorporaram algumas técnicas e hipóteses fundamentais da Inteligência Artificial como as hipóteses de sistemas físicos simbólicos, busca heurística e espaço de problemas.

Para colocarmos estas idéias em programas, não é obrigatório o uso de máquinas Xerox, Symbolics, ou estações Appolo, Sun e Prolog. Ou seja, para ser inteligente, um programa não precisa ser desenvolvido, necessariamente, em linguagens lógicas ou funcionais, mas sim apresentar uma implementação eficiente e satisfazer as hipóteses fundamentais da Inteligência Artificial conduzindo a solução efetiva de problemas.

Conforme afirmação de Bobrow [1985], a questão principal não é se Prolog ou outra linguagem qualquer é boa ou ruim , mas sim, como podemos, eficientemente, combinar técnicas e paradígmas, de maneira apropriada, de modo a resolvermos eficientemente os problemas.

Capítulo V
Desenvolvimento de uma Base de Conhecimento para o Sistema

Neste capítulo, faremos a descrição da base de conhecimento desenvolvida para o sistema SIGT, ressaltando a representação do conhecimento estático e declarativo, referente aos objetos e entidades do sistema ferroviário, e o conhecimento heurístico e imperativo, referente ao processo decisório responsável pelo gerenciamento e controle de trens.

5.1 - Níveis de Conhecimento

Segundo H. Simon [1981], o computador é um artefato, ou, mais precisamente, um sistema físico simbólico que visa a realização de objetivos (solução de problemas) através de uma simulação abstrata e racional (imitação) da conduta de sistemas naturais.

Esta simulação é possível, através da organização de sistemas físicos (que exibem comportamento quase idênticos aos naturais) em sistemas artificiais. Estes sistemas são produtos da síntese ou projeto, e do artifício , sendo caracterizados por objetivos, funções e adaptações ao meio externo .

Para tanto, carregam uma quantidade de conhecimento necessária a realização de seus objetivos e desempenho de suas funções. Este conhecimento é descrito de forma organizada no trabalho de A. Newell [1982], do qual traçaremos algumas observações importantes. A idéia básica é tentarmos definir os tipos de conhecimento necessários para que um sistema ,produto de um artefato racional (sistema físico simbólico), tenha conduta inteligente no processo de solução de problemas.

5.1.1 - Níveis Físico e Simbólico

Entre os níveis de conhecimento descritos por A. Newell, destacamos ,inicialmente, o conhecimento situado no nível simbólico.

Segundo M.S. Fox e J. McDermott [1986] este nível de conhecimento esta relacionado a capacidade de processamento requerida para resolver um problema observando as restrições de tempo e espaço. Os objetos de interesse deste nível estão voltados para as estruturas de

representação do conhecimento, bem como para os mecanismos de acesso e manutenção destas estruturas.

Abaixo do nível simbólico, Newell destaca os níveis de conhecimento relacionados com o "hardware" do computador incluindo o nível de dispositivos, o nível dos circuitos, e o nível de lógica compreendendo dois sub-níveis principais: nível de registros de transferência e nível de lógica de circuitos. Cada nível pode ser definido em termos de um conjunto de aspectos incluindo: objeto, componentes, leis de composição e conduta .

Estes níveis de conhecimento representam os níveis de um sistema de computador. Como exemplo, vamos mostrar, no quadro 5, a definição dos seguintes níveis, para um sistema de computador, segundo Newell:

quadro 5: níveis de conhecimento associados ao "hardware":

Aspectos	Nível registros/transferência	Nível simbólico
sistema	sistemas digitais	computadores
objeto	vetores de bits	símbolos, expressões
componentes	registradores e unidades funcionais	memória e operações
leis de composição	conexões entre componentes	atribuição e associação
conduta	operações lógicas	interpretação sequencial

Ademais, a existência destes níveis permitem definir o computador como um artifício que compartilha certas características organizacionais dos sistemas físicos simbólicos, sendo capazes de interpretar e executar estruturas simbólicas, que designam processos, que determinam operações sobre novas estruturas simbólicas, que designam, por sua vez, novos processos e assim sucessivamente.

5.1.2 - Hipótese do Nível Conhecimento

Segundo A. Newell [1982], existe um nível de conhecimento acima

do nível simbólico que é caracterizado pelo conhecimento como objeto e tem como lei de conduta o princípio da racionalidade.

Para M.S. Fox e J. McDermott [1986], este nível está relacionado ao tipo de conhecimento necessário a resolução de um problema, e não a forma como este conhecimento e efetivamente utilizado. Neste sentido, é dado uma importância maior ao conteúdo do conhecimento do que a forma propriamente dita.

A formação do nível conhecimento é caracterizada pelos seguintes aspectos:

I - Um agente (sistema no nível conhecimento) que compreende os seguintes componentes:

1- um corpo físico na forma de um conjunto de ações que possibilita ao mesmo atuar;

2- um corpo de conhecimento na forma de uma memória sem restrições e

3- um conjunto de objetivos.

Como um agente possui sempre estes componentes, não existem leis de composição para o nível conhecimento.

II - Uma lei de conduta que governa o agente (a conduta depende do que o agente sabe, do que ele quer e de que maneira ele interage físicamente com o ambiente) segundo o princípio da racionalidade, que pode ser descrito na forma:

" Se um agente tem conhecimento de que uma possível ação conduz a um dos objetivos, então o agente selecionará esta ação para executar."

Podemos fazer um paralelo do agente responsável pelo nível conhecimento com o agente econômico do trabalho de H. Simon [1981], relativo ao princípio da racionalidade econômica , o qual conduz a obtenção de soluções satisfatórias em lugar de soluções otimizantes em problemas de administração e gerenciamento de empresas.

Newell [1982] descreve, também,um princípio auxiliar que garante a equidade de ações aceitáveis, ou seja:

" Para um dado conhecimento, se duas ações conduzem para o mesmo objetivo, então ambas são selecionadas."

A respeito da satisfação simultânea de vários objetivos, Newell[1982] descreve outro princípio auxiliar onde afirma:

" Para um dado conhecimento se um objetivo G1 tem um conjunto de ações selecionadas A e o objetivo G2 tem um conjunto de ações selecionadas B, então o conjunto de ações preferenciais será formado pela interseção dos conjuntos A e B."

Este princípio auxiliar é semelhante ao princípio de agregação proposto por R.E. Bellman e L.A. Zadeh[1970], o qual determinou, no capítulo III de nosso trabalho, a elaboração de uma formulação multi-objetiva "fuzzy" para o problema de planejamento e "scheduling".

Nesta formulação, fizemos uso do operador "min" para representarmos a interseção lógica de conjuntos "fuzzy" associados aos objetivos do problema .

A respeito da conduta racional por parte dos agentes, E. A. Rich [1986] destaca cinco princípios ou processos fundamentais, que implicam no emprego de racionalidade e criatividade no sentido de escolher ações de modo a satisfazer um conjunto de objetivos.

1 - Inferência racional, através do uso de técnicas de representação do conhecimento e métodos de raciocínio;

2 - Ação racional, atendendo ao princípio da racionalidade. Neste caso, o processo de construção de uma sequência de ações, ou planejamento, envolve um procedimento de busca que deve empregar, cada vez mais, um método independente do domínio, ou de conhecimento intensivo, na tentativa de alcançar os objetivos;

3 - Busca racional, conforme análise anterior, devemos enfatizar o uso de heurísticas ou conhecimento, de modo a minimizar o efeito combinatório ou a explosão exponencial deste processo, através da redução do espaço de estados do problema;

4 - Satisfação racional, no sentido de realizar o menor esforço possível para atingir um conjunto de objetivos. Neste caso, o conflito surgido pela necessidade de um agente satisfazer, simultâneamente, um conjunto de objetivos sem muito esforço, conduz-nos ao processo de solução de problemas denominado satisfação racional de H. Simon [1981].

Neste processo, o agente não se preocupa em maximizar algum tipo

de função ou determinar a melhor situação associada a algum dos objetivos; mas sim, em satisfazer , para todos os objetivos, certos níveis de aspiração considerados satisfatórios. Este princípio, é compatível com o desenvolvimento das funções de pertinência existentes no SIGT e associadas aos objetivos do problema de planejamento e "scheduling" de trens.

5 - Estimativa ou suposição racional, ou seja, na falta de alguma informação factual necessária a realização do processo de planejamento, ao invés de interrompermos o processo , realizamos uma suposição baseada nos fatos conhecidos, minimizando a probabilidade de erros.

Quanto a criatividade necessária, principalmente, em atividades de projeto, E. A. Rich não faz distinção alguma com a forma racional de resolver problemas. Para Rich, uma ação criativa é fruto de uma heurística bem sucedida, o que não deixa de ser, a princípio, uma conduta racional.

III - Um objeto que constitui o próprio conhecimento:

Um agente trabalha, em princípio, através de um sistema simbólico, que possui a capacidade de atuar como se tivesse conhecimento próprio. Entretanto, a capacidade de atuar, segundo o princípio da racionalidade, requer o estabelecimento de um nível de conhecimento incluindo, um conhecimento hipotético e um conjunto de objetivos. Desta forma, o conhecimento é caracterizado, funcionalmente, em termos do que faz, e não estruturalmente, como os objetos dos níveis simbólicos e físicos do computador.

Durante a execução de um programa, condicionado, por exemplo, a geração e seleção de ações para atingir objetivos, determinando um espaço de estados, podemos dizer que o conhecimento é criado dinâmicamente, a medida em que nos aproximamos ,inteligentemente, dos objetivos. Portanto, alguma forma de conhecimento que conduza de forma inteligente a estes objetivos são objetos do nível conhecimento. As estruturas de representação (estados) e os mecanismos de acesso e manipulação (operadores) destas estruturas são objetos do nível simbólico que possibilitam realizar o conhecimento situado no nível imediatamente acima.

5.1.3 - Nível Organizacional

Segundo M.S. Fox e J. McDermott [1986] , uma conduta mais inteligente na solução de problemas necessita da atuação de vários agentes. Neste caso , é necessário definir um nível mais alto de conhecimento, denominado nível organizacional. Este nível está relacionado a forma como múltiplos agentes, com conhecimento limitado, podem cooperar na solução de problemas característicos de sistemas descentralizados.

5.2 - Conhecimento aplicado ao Planejamento e "Scheduling"

Visto que o emprego de conhecimento implica na efetiva utilização do princípio da racionalidade como lei de conduta, vamos tentar descrever os principais tipos de conhecimento necessários a realização das funções de planejamento e "scheduling", de forma racional ou explicativa, principalmente, em problemas do tipo "job-shop scheduling".

5.2.1 - Sistema ISIS

O sistema ISIS (M.S. Fox [1983]) incorpora um conjunto de restrições que permitem a realização de uma busca direcionada na programação e controle de tarefas ou ordens de serviço (sequência de operações que determinam a produção de algum tipo de produto) . Estas restrições incluem :

- cumprimento de objetivos;
- cumprimento de níveis de produção;
- atendimento a horários (hora inicial, tempo crítico, hora de término, etc);
- minimização de custos operacionais;
- atendimento a disponibilidade de recursos e ordens de serviço;
- produtividade de máquinas;
- garantia da qualidade e estabilidade do serviço;
- casualidades, como quebras e avarías de máquinas e interrupções de serviço;

. manutenção de máquinas;

. atendimento a preferências como prioridade de ordens de serviço e

. capacidade de operar com esquemas alternativos de máquinas e pessoal.

O sistema ISIS, dinâmicamente, resolve quais restrições deve utilizar e como utilizar no processo de programação e controle de tarefas, de modo a produzir "schedules" satisfatórios. A representação deste conhecimento, incluindo estruturas e processos, é feita com o uso de "schemas" ou "frames" através da linguagem SRL (M.S.Fox [1982]).

No nível simbólico, o conhecimento é organizado em um banco de dados que contem informações sobre a planta (máquinas, "lay-out", equipamento, componentes, esquemas de operação pessoal e restrições) responsável pela produção.

O processo de busca é dividido em 4 níveis: seleção de ordens de serviço, analise de capacidade, analise de recursos e, finalmente, distribuição de recursos.

Em recente trabalho, R.M.Kerr e R.V. Ebsary [1986], descrevem com detalhe a implementação de um sistema especialista para a programação de "schedules" ou para determinar a sequência ideal de operações ou transformações de produtos em um conjunto de máquinas.

Este sistema reduz a representação do conhecimento especialista a um banco de dados relacional (estrutura) que é operado por uma base de regras (processos), que contém basicamente:

. restrições a serem observadas durante a busca de modo a produzir um "schedule" satisfatório;

. meta-conhecimento que determina a estratégia de busca, e

. um esquema de relaxação de restrições.

As regras, em geral, são divididas em:

. regras que determinam prioridades em ordens de serviço;

. regras de precedência que determinam restrições na ordem na qual serviços e operações devem ser executados;

. regras que observam a disponibilidade de recursos, assegurando a viabilidade de "schedules";

. regras relacionadas a contigências casuais, como quebra e avaria de máquinas e

. regras que asseguram o cumprimento de metas e horários.

5.2.2 - Sistema SIGT

No sistema SIGT a realização de planejamento e "scheduling", ou seja, a elaboração de planos de horários para o gerenciamento do tráfego de trens atende a todas restrições descritas anteriormente. O conhecimento necessário a solução do problema é representado pelo desenvolvimento de um sistema físico simbólico , caracterizado por uma estrutura de representação de estados e mecanismos de manipulação (operadores), e de uma base de de conhecimento constituída de "frames".

A conduta racional é garantida pela realização de uma busca heurística, orientada segundo as restrições estruturais e flexíveis (objetivos) do problema , atendendo, também, um conjunto de regras ou produções, que permite a escolha inteligente e satisfatória de operadores.

5.3 - Base de Conhecimento associada ao Sistema

Através do desenvolvimento de uma base de conhecimento e de um sistema de produção, conseguimos a solução de problemas de "scheduling" multi-objetivos com restrições de capacidade, onde o número de possíveis soluções é da ordem de 2^{40}, em aproximadamente 35 seg., utilizando um processador INTEL 80386SX. Este tempo de resposta, é compatível com a aplicação afim , assegurando ,portanto, a viabilidade de implantação do sistema de controle e gerenciamento em tempo real.

Para a representação do conhecimento declarativo , factual e quase estático , representativo dos objetos, entidades e respectivos relacionamentos do sistema de operação ferroviário, foram definidas três "frames" principais que incorporam informações relativas a malha ferroviária (entidades), trens (objetos) e classes de trens (classe de objetos).

Quanto ao sistema de produção , que representa o conhecimento heurístico, imperativo e dependente do domínio, foram desenvolvidos 8

módulos de regras, organizados hierarquicamente em 3 níveis contendo , aproximadamente, 150 produções entre regras e meta-regras.

5.3.1 - "Frame" Classe de Trens

Seguindo a sintaxe de Modula-2, definimos a seguinte estrutura:

```
DEFINITION MODULE CLASSTREM;

(* especificação da interface *)

EXPORT QUALIFIED PtClass, CriaClass, ModificaPrioDin;

TYPE PtClass = POINTER TO ObjClass;

     ObjClass = RECORD
                  numero: CARDINAL;
                  nome: ARRAY [0..9] OF CHAR;
                  tempo: TipArray;
                  se,ss: CARDINAL;
                  PrioEst: CARDINAL;
                  PrioDin: CARDINAL;
                  comprimento: CARDINAL;
                  potencia: CARDINAL;
                  SecoesParada: TipCard
                END;

PROCEDURE CriaClass(cl:ARRAY OF CHAR; c:TipArray; a,b,d,e,f,g:CARDINAL;
                    sp:TipCard):PtClass;
PROCEDURE ModificaPrioDin (VAR Pc: PtClass; d: CARDINAL);
PROCEDURE ModSlotClass( VAR Pc: PtClass; tex: ARRAY OF CHAR);
PROCEDURE PrSlotClass( Pc: PtClass; tex: ARRAY OF CHAR);
PROCEDURE GravaC( Pc: PtClass);

END CLASSTREM.

IMPLEMENTATION MODULE CLASSTREM;


(* implementação de procedimentos e métodos *)

END CLASSTREM.
```

Em nosso sistema, definimos uma classe de trens como a representação de um conjunto de trens com composição constante e pré-definida, possuindo mesmo itinerário (origem, destino e pontos de parada) e tipo de carga.

Desta forma, podemos verificar a existência na "frame" de um conjunto de "slots" representado as principais informações

características de uma classe , incluíndo: número e nome da classe, tempos de percurso , prioridade estática, prioridade dinâmica (em circulação), comprimento e peso do trem e pontos de parada.

Todas as informações acima, com exceção da prioridade dinâmica, são invariáveis ao longo de um planejamento,só podendo serem modificadas através de mecanismos de acesso e manipulação (ModSlotClass e PrSlotClass) que modificam diretamente os "slots" (campos) de instâncias da classe e são importados por um módulo que controla a manipulação da base de conhecimento, e o processo de explanação de raciocínio.

Entretanto, a informação relativa a prioridade dinâmica, que caracteriza a performance de trens, é uma informação que pode variar sempre que houver necessidade de replanejamento. Portanto, é permitida a modificação deste valor através do uso de um procedimento que é importado por outros módulos do programa. O procedimento GravaC é utilizado na manutenção de arquivos, gravando as informações atribuídas as instâncias desta "frame".

5.3.2 - "Frame" Trens

Na sintaxe de Modula-2,temos a definição da seguinte estrutura:

```
DEFINITION MODULE OBJTREM;
(* especificação da interface *)
FROM CLASSTREM IMPORT PtClass;

EXPORT QUALIFIED PtTrem, CriaTrem, RetornaTempos, sentido, DestroiTrem,
prioridade, RetornaTempo, RetornaSecFinal, codigo, comprimento,
potencia, parada, SuperClass, TempoEntrada, RetornaSecInicial, GravaT,
NumClass, ModSlotTrem, PrSlotTrem, PrioDinamica, RetornaSecInicialClass,
RetornaSecFinalClass;

TYPE  PtTrem;      (* tipo opaco *)

PROCEDURE CriaTrem (pt: PtClass; t: REAL; s: CHAR; c,c1,c2: CARDINAL ):
PtTrem;
PROCEDURE RetornaTempos (Pt: PtTrem; VAR t: TipArray);
PROCEDURE sentido(Pt: PtTrem): CARDINAL;
PROCEDURE SuperClass(Pt: PtTrem; VAR nome: ARRAY OF CHAR);
PROCEDURE prioridade(Pt: PtTrem): CARDINAL;
PROCEDURE PrioDinamica(Pt: PtTrem): CARDINAL;
PROCEDURE TempoEntrada(Pt: PtTrem): REAL;
```

```
PROCEDURE RetornaTempo(Pt: PtTrem; s: CARDINAL): REAL;
PROCEDURE RetornaSecFinal( Pt: PtTrem): CARDINAL;
PROCEDURE RetornaSecInicial(Pt:PtTrem): CARDINAL;
PROCEDURE comprimento(Pt: PtTrem): CARDINAL;
PROCEDURE RetornaSecFinalClass(Pt: PtTrem): CARDINAL;
PROCEDURE RetornaSecInicialClass(Pt:PtTrem): CARDINAL;
PROCEDURE codigo(Pt: PtTrem): CARDINAL;
PROCEDURE potencia(Pt: PtTrem): CARDINAL;
PROCEDURE NumClass(Pt: PtTrem): CARDINAL;
PROCEDURE DestroiTrem(Pt: PtTrem);
PROCEDURE parada(Pt: PtTrem; p:CARDINAL): CARDINAL;
PROCEDURE ModSlotTrem( VAR Pt: PtTrem; tex: ARRAY OF CHAR);
PROCEDURE PrSlotTrem( Pt: PtTrem; tex: ARRAY OF CHAR);
PROCEDURE GravaT( Pt: PtTrem);

END OBJTREM.

IMPLEMENTATION MODULE OBJTREM;

FROM CLASSTREM IMPORT PtClass;

TYPE PtTrem = POINTER TO ObjTrem;

     ObjTrem = RECORD
                SuperClass: PtClass;
                TempoEntrada: REAL;
                sentido: CHAR;
                codigo: CARDINAL;
                SecaoEntrada, SecaoSaida: CARDINAL
               END;

(* implementação de procedimentos e métodos *)

END OBJTREM.
```

Desta forma, um trem fica caracterizado por uma classe, da qual herda as informações comuns, e um conjunto de "slots" que contém informações próprias, incluindo: a hora de entrada em circulação, o sentido (par ou impar), o código, bem como as seções de entrada e saída específicas do planejamento a ser realizado.

É interessante observarmos, que as informações providas pelos "slots" próprios ou herdados, são somente acessadas por meio de métodos (funções), garantindo a integridade dos dados. Foram definidos, também , a exemplo da "frame" classe de trens, procedimentos para criação de instâncias, manipulação, acesso e manutenção de arquivos. Permitimos, também, a destruição de instâncias já que trens no nosso sistema constituem objetos dinâmicos com duração limitada.

Para definirmos instâncias de classes de trens e de trens,

realizamos a seguinte implementação, na sintaxe de Modula-2:

```
DEFINITION MODULE DEFTREM;
(* especificação da interface *)
FROM SIMTREM IMPORT SimulaTempos, SimulaClass, inicializa, NumTrem;

FROM OBJTREM IMPORT PtTrem, CriaTrem, DestroiTrem;

FROM CLASSTREM IMPORT PtClass, CriaClass;;

EXPORT QUALIFIED trem, class, DefineTrens, DefineClasses, InsereTrem,
DeleteTrem, ntt, ntc, SimulaTrens, GravaTrem, GravaClass;

VAR trem: ARRAY [1..16] OF PtTrem;
    nti, ntt, ntc: CARDINAL;
    class: ARRAY [1..10] OF PtClass;

PROCEDURE DefineTrens( nt: CARDINAL );
PROCEDURE SimulaTrens( nt: CARDINAL );
PROCEDURE DeleteTrem(n: CARDINAL);
PROCEDURE InsereTrem(pc,se,ss: CARDINAL; s: CHAR; t: REAL);
PROCEDURE DefineClasses(nc, cod: CARDINAL);
PROCEDURE GravaTrem();
PROCEDURE GravaClass();
END DEFTREM.

IMPLEMENTATION MODULE DEFTREM;

(* implementação dos procedimentos *)

END DEFTREM;
```

Os procedimentos DefineTrens e DefineClasses são responsáveis pela instanciação e inicialização dos objetos (instâncias terminais e variáveis de classe) através do preenchimento dos "slots" das respectivas "frames".

As instâncias de classes de trens e de trens estão organizadas segundo um mapeamento finito de ponteiros , permitindo o respectivo acesso direto e facilitando as operações de inserção e deleção de objetos. Oferecemos, também, a possibilidade de simularmos a criação de classes associando as mesmas uma tabela de frequências e tempos de percurso aleatórios.

Desta forma, é possível simular para cada trem a classe pertencente segundo a tabela de frequências, hora de entrada e sentido de movimento. Neste caso, as seções de entrada e saida são herdadas da classe respectiva.

5.3.3 - "Frame" Seções

No nosso sistema, uma malha ferroviária sob planejamento é definida como uma sequência de seções enumeradas. Para efeito da circulação de trens, as seções podem ser classificadas como seções de linha dupla e seções de linha singela, ambas permitindo o tráfego bidirecional e possuindo um comprimento próprio. No sistema de controle de tráfego centralizado, é possível acumular trens dentro de uma mesma seção (singela ou na mesma linha de uma seção dupla) quando os mesmos trafegam em um mesmo sentido. Para tanto, precisamos definir o número de blocos (cada bloco comporta um trem) que comporta a seção e o tempo de "headway" (afastamento mínimo entre trens que trafegam no mesmo sentido).

Nas linhas singelas, a interface entre seções é determinada pelos pátios de cruzamentos ou desvios (pontos onde é possível o cruzamento de trens) . Estes pátios são caracterizados pelo respectivo tamanho, capacidade (número de linhas) e valores de resistência a demarragem de trens. Já, nas linhas duplas, a interface é caracterizada pelos travessões (pontos onde os trens podem trocar de linhas). Tanto as entradas de pátios como os pontos de travessoes são controlados remotamente de um centro localizado por chaves que permitem a mudança no movimento dos trens (mudança de linha).

Outra informação importante a respeito das seções se refere ao "status" ou a situação das linhas e pátios de cruzamento. Neste particular, podemos considerar 3 tipos de situações:

1 - desativado : quando o pátio está desativado ou impedido, não é permitida a parada ou passagem de outro trem, até que o mesmo seja ativado. Quando uma linha esta desativada ou impedida, não é permitida a circulação de trens. Caso a linha pertença a uma seção dupla a seção se comporta como uma seção singela. Entretanto, se a linha pertence a uma seção singela o tráfego de trens nesta seção fica totalmente impedido;

2 - desocupado: neste caso, para pátios é permitido a parada ou passagem de trens, bem como a circulação, tratando-se de linhas;

3 - ocupado: neste caso, a parada ou passagem de trens em pátios

fica, temporariamente, proibida até que haja a desocupação. O mesmo é válido para a circulação de trens em linhas.

Para as seções singelas, interessam as informações dos dois pátios que definem as interfaces, ao qual denominamos, respectivamente, de pátio ímpar e pátio par (não admitiremos impedimento total de tráfego). Para as seções duplas, interessam o status das linhas segundo o sentido de movimento ímpar ou par. Para distinguirmos as linhas adotamos a nomenclatura de linha principal e linha secundária.

Desta forma, definimos a seguinte estrutura, na sintaxe de Modula-2:

```
DEFINITION MODULE OBJSECAO;
(* especificação da interface *)
EXPORT QUALIFIED PtSecao, CriaSecaoSingela, CriaSecaoDupla,
ModificaStatus, NumLinhas, CapacPar, CapacImp, RestPar, RestImp,
StatusPar, StatusImp, numero, StatusPriPar, StatusPriImp, StatusSecPar,
StatusSecImp, DestroiSecao, ModSlotSecao, PrSlotSecao, NomeSecao,
HeadwayInicial, HeadwayFinal, TipStatus, GravaS, RestauraStatus,
comprimento;

TYPE PtSecao;  ( tipo opaco )

     TipStatus = (vazio, desativada, ocupada, desocupada);

PROCEDURE CriaSecaoSingela (s1,s2: TipStatus; n,n1,r1,r2,c1,c2,c3,
          c4,c5,c6: cardinal; h1,h2: REAL; s: ARRAY OF CHAR): PtSecao;
PROCEDURE CriaSecaoDupla (s1,s2,s3,s4: TipStatus; n,n1,c3,c4: CARDINAL;
                          h1,h2: REAL;s: ARRAY OF CHAR): PtSecao;
PROCEDURE NumLinhas( Pt: PtSecao): CARDINAL;
PROCEDURE comprimento( Pt: PtSecao): CARDINAL;
PROCEDURE CapacPar( Pt: PtSecao): CARDINAL;
PROCEDURE CapacImp( Pt: PtSecao): CARDINAL;
PROCEDURE RestPar(Pt: PtSecao): CARDINAL;
PROCEDURE RestImp(Pt: PtSecao): CARDINAL;
PROCEDURE StatusPar (Pt: PtSecao): CARDINAL;
PROCEDURE StatusImp (Pt: PtSecao): CARDINAL;
PROCEDURE numero (Pt: PtSecao): CARDINAL;
PROCEDURE StatusPriPar (Pt: PtSecao): CARDINAL;
PROCEDURE StatusPriImp (Pt: PtSecao): CARDINAL;
PROCEDURE StatusSecPar (Pt: PtSecao): CARDINAL;
PROCEDURE StatusSecImp (Pt: PtSecao): CARDINAL;
PROCEDURE HeadwayFinal (Pt: PtSecao): REAL;
PROCEDURE HeadwayInicial (Pt: PtSecao): REAL;
PROCEDURE ModificaStatus (VAR Pt: PtSecao; tag: CARDINAL);
PROCEDURE RestauraStatus (VAR Pt: PtSecao; tag: CARDINAL);
PROCEDURE DestroiSecao (Pt: PtSecao);
PROCEDURE ModSlotSecao(VAR Pt: PtSecao; tex: ARRAY OF CHAR);
PROCEDURE PrSlotSecao(Pt: PtSecao; tex: ARRAY OF CHAR);
PROCEDURE NomeSecao(Pt: PtSecao; VAR ch: ARRAY OF CHAR);
```

```
PROCEDURE GravaS(Pt: PtSecao);

END OBJSECAO.

IMPLEMENTATION MODULE OBJSECAO;

TYPE PtSecao = POINTER TO ObjSecao;

     TipLinha = (singela,dupla);

     ObjSecao = RECORD
                  nome: TipNome;
                  NumLinhas,numero, comprimento, NumBlocos: CARDINAL;
                  HeadFim, HeadIni: REAL;
                  CASE tag: TipLinha OF
        singela :   RestPar, RestImp: CARDINAL;
                    CapacPar, CapacImp: CARDINAL;
                    DesvPar, DesvImp: CARDINAL;
                    StatusPar, StatusImp: TipStatus;
        dupla   :   StatusPriPar, StatusPriImp,
                    StatusSecPar, StatusSecImp: TipStatus
                  END
                END;


 (* implementação de procedimentos e métodos *)

 END OBJSECAO.
```

Além do conjunto de métodos, foram implementados também , a exemplo da "frame" trem, procedimentos para criação e destruição de instâncias, manipulação, acesso, manutenção de arquivos, e para modificação do status dos pátios e linhas (informação dinâmica).

Para definirmos instâncias de seções, realizamos a seguinte implementação, na sintaxe de Modula-2:

```
 DEFINITION MODULE DEFSECAO;
 (* especificação da interface *)
 FROM OBJSECAO IMPORT PtSecao, CriaSecaoSingela, CriaSecaoDupla,
 DestroiSecao, TipStatus, GravaS;

 EXPORT QUALIFIED  secao, DefineSecoes, DeleteSecao, InsereSecao, si,sf,
 GravaSecoes;

 VAR secao: ARRAY [1..24] OF PtSecao;

     si,sf: CARDINAL;

 PROCEDURE DefineSecoes(ni,ns: CARDINAL);
 PROCEDURE InsereSecao(n: CARDINAL);
 PROCEDURE DeleteSecao(n: CARDINAL);
```

```
PROCEDURE GravaSecoes();

END DEFSECAO.

IMPLEMENTATION MODULE DEFSECAO;
(* implementação dos procedimentos *)
END DEFSECAO.
```

Da mesma forma, as instâncias de seções estão organizadas segundo um mapeamento finito de ponteiros , permitindo o respectivo acesso direto e facilitando as operações de alteração da malha ferroviária através da inserção e deleção de objetos (instâncias terminais de seções).

5.4 - Organização das Regras de Produção

5.4.1 - Tipos de Regras

Entre os tipos principais de regras desenvolvidas para o SIGT podemos destacar:

. meta-regras : regras que controlam o conhecimento;

. restrições : regras que restrigem o espaço de estados do problema , analisando a viabilidade de operadores quanto a satisfação de restrições físicas e operacionais do problema ;

. regras de análise: regras que analizam o tipo de conflito;

. regras de conflito : regras que determinam a solução de conflitos e, consequentemente, a atribuição de preferência a operadores;

. regras de planejamento : regras que controlam o progresso dos planos e que determinam, quando necessário, a reelaboração de planos e reavaliação de prioridades ;

. regras de despacho ou ações: regras que determinam a tomada de decisões observando a viabilidade das ações programadas;

. regras heurísticas : regras que incorporam o conhecimento especialista no reconhecimento de situações e na solução de conflitos, proporcionando a realização de uma busca inteligente e

. regras estratégicas : regras que introduzem estratégias que determinam prioridades na solução de conflitos ou alteram os parâmetros associados aos objetivos do problema.

O conhecimento associado a estas regras possibilita ao sistema

de produção desempenhar as seguintes funções principais :

. garantir a viabilidade de restrições físicas e restrições operacionais;

. prevenir a ocorrência de bloqueios (conflitos sem soluções)

. possibilitar uma busca inteligente através do uso de heurísticas e

. reduzir o espaço de estados do problema através da resolução de impasses e na determinação de preferência em operadores.

O sistema de produção desenvolvido para o SIGT constitui um modelo hierárquico que abrange três níveis principais : nível de análise, nível decisório e um nível de restrição, como veremos a seguir.

5.4.2 - Nível de Análise

Este nível, mais alto da hierarquia, consiste de um módulo de regras denominado RuleSet0. Este módulo é responsável pela análise do tipo de conflito, e, também, através do uso de meta-regras apropriadas, responsável pela ativação do módulo de regras responsável pela determinação de preferências. Na sintaxe de Modula-2, temos:

```
DEFINITION MODULE RuleSet0;
(* especificação da interface *)
EXPORT QUALIFIED preferencia;

PROCEDURE preferencia (PosSecao, t1, t2, cr: CARDINAL): CARDINAL;

END RuleSet0.
```

Observando a especificação do modulo, verificamos que o mesmo é representado funcionalmente pela função "preferencia", que recebe como parâmetros os trens envolvidos no conflito , a respectiva seção da ferrovia onde o conflito se realizaria e um valor que regula o mecanismo de explanação de raciocínio. Esta função devolve um valor do tipo CARDINAL associado aos possíveis valores oper1, oper2, oper3, oper4, oper5, oper6 e oper7. Estes valores, que são atribuidos ao objeto 'objetivo', representam possíveis preferências em operadores.

Em seguida, temos a implementação do módulo RuleSet0:

```
MPLEMENTATION MODULE RuleSet0;

FROM INTERPRETE IMPORT When, And, Then, InicializaLoop, IncLoop,
IniciaPilha, maior, menor, igual, EsvaziaPilha;

FROM DEFTREM IMPORT trem;

FROM OBJTREM IMPORT sentido, prioridade;

FROM WorkMemory IMPORT objeto, faca, remova, cria, atributo,
ModificaAtributo, PrintAtributo, LegValConflito, LegValSent, LegValPref;

FROM DEFSECAO IMPORT secao;

FROM OBJSECAO IMPORT NumLinhas;

FROM RuleSet1 IMPORT PrefParar;

FROM RuleSet2 IMPORT PrefAvancar;

FROM RuleSet5 IMPORT PrefEncostar;

FROM RuleSet6 IMPORT PrefUltrapassar;

FROM RuleSet7 IMPORT PrefAguardar;

FROM EXPLAIN IMPORT explain;

PROCEDURE preferencia (PosSecao, t1, t2, cr: CARDINAL): CARDINAL;
VAR ContLoop, aux: CARDINAL;
    EstoraLoop: BOOLEAN;
    conflito, objetivo: objeto;
BEGIN
  IniciaPilha();
  ContLoop:= 0;
  EstoraLoop:= FALSE;
  InicializaLoop(1,1,1,7,36,38,1,TRUE);

  cria(conflito," conflito",0);
  cria(objetivo," objetivo",0);

  REPEAT

    When (sentido(trem[t1]),igual,ORD(par),1,36);
      And (sentido(trem[t2]),igual,ORD(impar),1);
    Then (faca,conflito,1,ORD(cruzamento),36);

    When (sentido(trem[t1]),igual,ORD(impar),1,37);
      And (sentido(trem[t2]),igual,ORD(par),1);
    Then (faca,conflito,1,ORD(cruzamento),37);

    When (sentido(trem[t1]),igual,sentido(trem[t2]),1,38);
      Then (faca,conflito,1,ORD(ultrapassagem),38);

    When (atributo(1,conflito),igual,ORD(cruzamento),1,1);
      And (NumLinhas(secao[PosSecao]),igual,1,1);
    Then (faca,objetivo,1,PrefParar(PosSecao,t1,t2),1);
```

```
    When (atributo(1,conflito),igual,ORD(cruzamento),1,2);
      And (NumLinhas(secao[PosSecao]),maior,1,1);
    Then (faca,objetivo,1,PrefAvancar(PosSecao,t1,t2),2);

    When (atributo(1,conflito),igual,ORD(ultrapassagem),1,3);
      And (NumLinhas(secao[PosSecao]),igual,1,1);
      And (prioridade(trem[t1]),maior,prioridade(trem[t2]),1);
    Then (faca,objetivo,1,PrefEncostar(PosSecao,t1,t2),3);

    When (atributo(1,conflito),igual,ORD(ultrapassagem),1,4);
      And (NumLinhas(secao[PosSecao]),igual,1,1);
      And (prioridade(trem[t1]),menor,prioridade(trem[t2]),1);
    Then (faca,objetivo,1,PrefEncostar(PosSecao,t1,t2),4);

    When (atributo(1,conflito),igual,ORD(ultrapassagem),1,5);
      And (NumLinhas(secao[PosSecao]),maior,1,1);
      And (prioridade(trem[t1]),maior,prioridade(trem[t2]),1);
    Then (faca,objetivo,1,PrefUltrapassar(PosSecao,t1,t2),5);

    When (atributo(1,conflito),igual,ORD(ultrapassagem),1,6);
      And (NumLinhas(secao[PosSecao]),maior,1,1);
      And (prioridade(trem[t1]),menor,prioridade(trem[t2]),1);
    Then (faca,objetivo,1,PrefUltrapassar(PosSecao,t1,t2),6);

    When (atributo(1,conflito),igual,ORD(ultrapassagem),1,7);
      And (prioridade(trem[t1]),igual,prioridade(trem[t2]),1);
    Then (faca,objetivo,1,PrefAguardar(PosSecao,t1,t2),7);

    IncLoop(ContLoop,EstoraLoop);
    aux:= atributo(1,objetivo);

  UNTIL EstoraLoop OR (aux > 0);

  remova(conflito);
  remova(objetivo);
  IF (cr = 1)  THEN
    explain(aux,PosSecao,t1,t2)
  END;
  EsvaziaPilha();
  RETURN aux

END preferencia;

END RuleSet0.
```

Na forma textual, as regras apresentam o seguinte conteúdo:

módulo (ruleset 0):

abstração funcional (objetivo): determine preferência;

preferências (valores legais): oper1, oper2, oper3, oper4, oper5, oper6 e oper7;

importa (meta-regras): preferência para parar, preferência para avançar, preferência para encostar, preferência para ultrapassar e preferência para aguardar (manter "headway");

produções:

```
regra (036)
Quando (sentido do trem1 é par)
E (sentido do trem2 é ímpar)
Então (conflito é do tipo cruzamento)

regra (037)
Quando (sentido do trem1 é ímpar)
E (sentido do trem2 é par)
Então (conflito é do tipo cruzamento)

regra (038)
Quando (sentido do trem1 é igual a sentido do trem2)
Então (conflito é do tipo ultrapassagem)

regra (001)
Quando (conflito é do tipo cruzamento)
E (seção e singela)
Então (determine preferência para parar)

regra (002)
Quando (conflito é do tipo cruzamento)
E (seção é dupla)
Então (determine preferência para avançar)

regra (003)
Quando (conflito é do tipo ultrapassagem)
E (seção é singela)
E (prioridade do trem1 é maior que prioridade do trem2)
Então (determine preferência para encostar trem2)

regra (004)
Quando (conflito é do tipo ultrapassagem)
E (seção é singela)
E (prioridade do trem1 é menor que prioridade do trem2)
Então (determine preferência para encostar trem1)

regra (005)
Quando (conflito é do tipo ultrapassagem)
E (seção é dupla)
E (prioridade do trem1 é maior que prioridade do trem2)
Então (determine preferência para ultrapassar trem2)

regra (006)
Quando (conflito é do tipo ultrapassagem)
E (seção é dupla)
E (prioridade do trem1 é menor que prioridade do trem2)
Então (determine preferência para ultrapassar trem1)

regra (007)
Quando (conflito é do tipo ultrapassagem)
E (prioridade do trem1 é igual prioridade do trem2)
Então (determine preferência para aguardar)
```

Conforme podemos verificar, as regras de número 1 a 7

representam meta-regras que acionam módulos de regras segundo o tipo de conflito.

5.4.3 - Nível Decisório

- Árvore de Decisões:

A arvore de decisoes na determinação de preferências, segundo o tipo de conflito entre dois trens em uma seção qualquer da ferrovia, e descrita, conforme a figura 38.

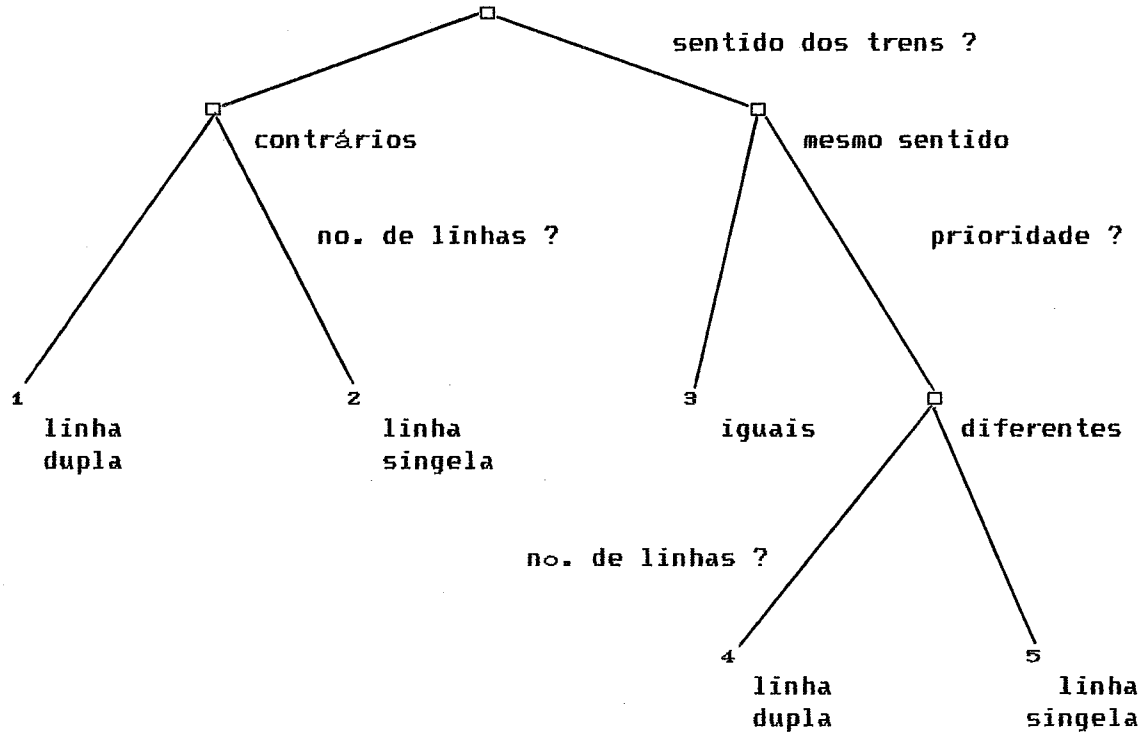


figura 38: árvore de decisões na solução de conflitos.

Desta forma , segundo o tipo de conflito, o sistema de produção , através das meta-regras iniciais , determina a chamada dos seguintes módulos de regras, situados no nível médio da hierarquia, possibilitando os seguintes tipos de preferências :

- Cruzamento em Linha Dupla (tipo 1):

módulo (ruleset 2):

abstração funcional (objetivo): determine preferência para trem avançar;

preferências:

oper1 - trem1 avança na linha secundária
oper2 - trem2 avança na linha secundária
oper4 - rejeição;

importa (meta-regras): condição de trem avançar;

produções:

regra (060)
Quando (prioridade do trem1 é maior que prioridade do trem2)
E (condição do trem1 avançar na principal é permitida)
E (condição do trem2 avançar na secundária é permitida)
Então (de preferência para avançar trem2 na linha secundária)

regra (061)
Quando (prioridade do trem1 é maior que prioridade do trem2)
E (condição do trem1 avançar na principal é não permitida)
E (condição do trem1 avançar na secundária é permitida)
E (condição do trem2 avançar na principal é permitida)
Então (de preferência para avançar trem1 na linha secundária)

regra (062)
Quando (prioridade do trem2 é maior que prioridade do trem1)
E (condição do trem2 avançar na principal é permitida)
E (condição do trem1 avançar na secundária é permitida)
Então (de preferência para avançar trem1 na linha secundária)

regra (063)
Quando (prioridade do trem1 é maior que prioridade do trem2)
E (condição do trem2 avançar na principal é não permitida)
E (condição do trem2 avançar na secundária é permitida)
E (condição do trem1 avançar na principal é permitida)
Então (de preferência para avançar trem2 na linha secundária)

regra (064)
Quando (prioridade do trem1 é igual a prioridade do trem2)
E (condição do trem1 avançar na principal é permitida)
E (condição do trem2 avançar na secundária é permitida)
Então (de preferência para avançar trem2 na linha secundária)

regra (065)
Quando (prioridade do trem1 é igual a prioridade do trem2)
E (condição do trem2 avançar na principal é permitida)
E (condição do trem1 avançar na secundária é permitida)
Então (de preferência para avançar trem1 na linha secundária)

regra (066)
Quando (condição do trem2 avançar na principal é não permitida)
E (condição do trem2 avançar na secundária é não permitida)
Então (rejeição)

regra (067)

Quando (condição do trem1 avançar na principal é não permitida)
E (condição do trem1 avançar na secundária é não permitida)
Então (rejeição)

regra (012)
Quando (condição do trem1 avançar na principal não é conhecida)
Então (determine condição do trem1 avançar na linha principal)

regra (013)
Quando (condição do trem1 avançar na secundaria não é conhecida)
Então (determine condição do trem1 avançar na linha secundária)

regra (014)
Quando (condição do trem2 avançar na principal não é conhecida)
Então (determine condição do trem2 avançar na linha principal)

regra (015)
Quando (condição do trem2 avançar na secundaria não é conhecida)
Então (determine condição do trem2 avançar na linha secundária)

- Cruzamento em Linha Singela (tipo 2):

módulo (ruleset 1):

abstração funcional (objetivo): determine preferência para trem parar;

preferências:

oper1 - trem1 para no cruzamento
oper2 - trem2 para no cruzamento
oper3 - não há preferência
oper4 - rejeição
oper7 - não há conflito;

importa (meta-regra): condição de trem parar;

produções:

regra (039)
Quando (trem2 já aguarda no pátio de cruzamento)
Ou (trem1 também já aguarda no pátio de cruzamento)
Então (não existe conflito de cruzamento)

regra (040)
Quando (condição de parar trem1 é obrigatória)
E (condição de parar trem2 é não permitida)
Então (de preferência para parar trem1)

regra (041)
Quando (condição de parar trem1 é permitida)
E (condição de parar trem2 é não permitida)
Então (de preferência para parar trem1)

regra (042)
Quando (condição de parar trem1 é não permitida)
E (condição de parar trem2 é obrigatória)

Então (de preferência para parar trem2)

regra (043)
Quando (condição de parar trem1 é não permitida)
E (condição de parar trem2 é permitida)
Então (de preferência para parar trem2)

regra (044)
Quando (condição de parar trem2 é obrigatória)
E (condição de parar trem1 é permitida)
Então (de preferência para parar trem2)

regra (045)
Quando (condição de parar trem1 é obrigatória)
E (condição de parar trem2 é permitida)
Então (de preferência para parar trem1)

regra (046)
Quando (condição de parar trem1 é permitida)
E (condição de parar trem2 é permitida)
E (prioridade do trem1 é igual prioridade do trem2)
Então (não há preferência)

regra (047)
Quando (condição de parar trem1 é obrigatória)
E (condição de parar trem2 é obrigatória)
Então (não há preferência)

regra (048)
Quando (condição de parar trem1 é não permitida)
E (condição de parar trem2 é não permitida)
Então (rejeição)

regra (049)
Quando (condição de parar trem1 é permitida)
E (condição de parar trem2 é permitida)
E (prioridade do trem1 é menor que prioridade do trem2)
E (performance do trem1 é aceitável)
Então (de preferência para parar trem1)

regra (050)
Quando (condição de parar trem1 é permitida)
E (condição de parar trem2 é permitida)
E (prioridade do trem1 é maior que prioridade do trem2)
E (performance do trem2 é aceitável)
Então (de preferência para parar trem2)

regra (051)
Quando (condição de parar trem1 é permitida)
E (condição de parar trem2 é permitida)
E (performance do trem1 é inaceitável)
E (performance do trem2 é aceitável)
Então (de preferência para parar trem2)

regra (052)
Quando (condição de parar trem1 é permitida)
E (condição de parar trem2 é permitida)
E (performance do trem2 é inaceitável)

E (performance do trem1 é aceitável)
Então (de preferência para parar trem1)

regra (053)
Quando (condição de parar trem1 é permitida)
E (condição de parar trem2 é permitida)
E (performance do trem1 é inaceitável)
E (performance do trem2 é inaceitável)
Então (rejeição)

regra (010)
Quando (condição de parar trem1 não é conhecida)
E (trem1 não aguarda em pátio de cruzamento)
Então (determine condição de trem1 parar na seção)

regra (011)
Quando (condição de parar trem2 não é conhecida)
E (trem2 não aguarda em pátio de cruzamento)
Então (determine condição de trem2 parar na seção)

- Manutenção do "Headway" (tipo 3):

módulo (ruleset 7):

abstração funcional (objetivo): determine preferência para trem aguardar;

preferências: se algum trem já aguarda na seção a frente então

oper1 - mantem trem1 na seção a frente (pátio posterior)
oper2 - mantem trem2 na seção a frente (pátio posterior)
oper3 - não há preferência

senão

oper1 - trem1 aguarda na seção (pátio anterior)
oper2 - trem2 aguarda na seção (pátio anterior)
oper3 - não há preferência
oper5 - retarda trem anterior na principal;

importa (meta-regra): condição de trem aguardar (parar);

produções:

regra (120)
Quando (trem2 já aguarda a frente)
E (trem1 também já aguarda a frente)
E (condição do trem1 aguardar antes é permitida)
Então (não há preferência para trem aguardar)

regra (121)
Quando (trem2 já aguarda a frente)
E (trem1 não aguarda a frente)
E (condição do trem1 aguardar antes é não permitida)
Então (de preferência para manter trem2 no pátio a frente)

regra (122)
Quando (trem1 já aguarda a frente)

E (trem2 não aguarda a frente)
E (condição do trem2 aguardar antes é permitida)
Então (não há preferência para trem aguardar)

regra (123)
Quando (trem1 já aguarda a frente)
E (trem2 não aguarda a frente)
E (condição do trem2 aguardar antes é não permitida)
Então (de preferência para manter trem1 no pátio a frente)

regra (124)
Quando (trem1 já aguarda a frente)
E (trem2 também já aguarda a frente)
Então (não há preferência para trem aguardar)

regra (125)
Quando (trem1 não aguarda a frente)
E (trem2 também não aguarda a frente)
E (condição do trem1 aguardar antes é obrigatória)
E (condição do trem2 aguardar antes não é obrigatória)
Então (de preferência para trem1 aguardar na seção)

regra (126)
Quando (trem1 não aguarda a frente)
E (trem2 também não aguarda a frente)
E (condição do trem1 aguardar antes é permitida)
E (condição do trem2 aguardar antes é permitida)
Então (não há preferência para trem aguardar)

regra (127)
Quando (trem2 não aguarda a frente)
E (trem1 também não aguarda a frente)
E (condição do trem2 aguardar antes é obrigatória)
E (condição do trem1 aguardar antes não é obrigatória)
Então (de preferência para trem2 aguardar na seção)

regra (128)
Quando (trem1 não aguarda a frente)
E (trem2 também não aguarda a frente)
E (condição do trem1 aguardar antes é permitida)
E (condição do trem2 aguardar antes é não permitida)
Então (de preferência para trem1 aguardar na seção)

regra (129)
Quando (trem2 não aguarda a frente)
E (trem1 também não aguarda a frente)
E (condição do trem2 aguardar antes é permitida)
E (condição do trem1 aguardar antes é não permitida)
Então (de preferência para trem2 aguardar na seção)

regra (130)
Quando (trem1 não aguarda a frente)
E (trem2 também não aguarda a frente)
E (condição do trem1 aguardar antes é obrigatória)
E (condição do trem2 aguardar antes é obrigatória)
Então (não há preferência para trem aguardar)

regra (131)

Quando (trem2 não aguarda a frente)
E (trem1 também não aguarda a frente)
E (condição do trem2 aguardar antes é não permitida)
E (condição do trem1 aguardar antes é não permitida)
Então (retarda trem1 na principal)

regra (026)
Quando (condição do trem1 aguardar antes não é conhecida)
Então (determine condição do trem1 aguardar na seção)

regra (027)
Quando (condição do trem2 aguardar antes não é conhecida)
Então (determine condição do trem2 aguardar na seção)

regra (028)
Quando (condição do trem2 aguardar depois não é conhecida)
E (sentido do trem2 é ímpar)
Então (determine condição do trem2 aguardar seção a frente sentido imp)

regra (029)
Quando (condição do trem1 aguardar depois não é conhecida)
E (sentido do trem1 é ímpar)
Então (determine condição do trem2 aguardar seção a frente sentido imp)

regra (030)
Quando (condição do trem2 aguardar depois não é conhecida)
E (sentido do trem2 é par)
Então (determine condição do trem2 aguardar seção a frente sentido par)

regra (031)
Quando (condição do trem1 aguardar depois não é conhecida)
E (sentido do trem1 é par)
Então (determine condição do trem2 aguardar seção a frente sentido par)

- Ultrapassagem em Linha Dupla (tipo 4):

módulo (ruleset 6):

abstração funcional (objetivo): determine preferência para trem ultrapassar;

preferências:

oper1 - trem prioritário ultrapassa pela principal
oper2 - trem prioritário ultrapassa pela secundária
oper3 - não há preferência
oper4 - rejeição
oper5 - retarda trem prioritário na principal
oper6 - retarda trem prioritário na secundária;

importa (meta-regra): condição de trem ultrapassar (avançar);

produções:

regra (106)
Quando (condição de avançar na principal é não permitida)

E (condição de avançar na secundária é não permitida)
Então (rejeição)

regra (107)
Quando (condição de avançar na principal é permitida)
E (condição de avançar na secundária é não permitida)
Então (retarda ultrapassagem pela principal)

regra (108)
Quando (condição de avançar na principal é não permitida)
E (condição de avançar na secundária é permitida)
Então (retarda ultrapassagem pela secundária)

regra (109)
Quando (condição de avançar na principal é permitida)
E (sentido dos trens é par)
E (seção a frente é singela)
E (seção a frente não permite avanço de trem par)
Então (retarda ultrapassagem pela principal)

regra (110)
Quando (condição de avançar na secundária é permitida)
E (sentido dos trens é ímpar)
E (seção a frente é singela)
E (seção a frente não permite avanço de trem ímpar)
Então (retarda ultrapassagem pela secundária)

regra (111)
Quando (condição de avançar na secundária é permitida)
E (sentido dos trens é par)
E (seção a frente é singela)
E (seção a frente não permite avanço de trem par)
Então (retarda ultrapassagem pela secundária)

regra (112)
Quando (condição de avançar na principal é permitida)
E (sentido dos trens é ímpar)
E (seção a frente é singela)
E (seção a frente não permite avanço de trem ímpar)
Então (retarda ultrapassagem pela principal)

regra (113)
Quando (condição de avançar na principal é permitida)
E (condição de avançar na secundária é permitida)
E (sentido dos trens é par)
E (seção a frente é singela)
E (seção a frente permite avanço de trem par)
Então (de preferência para ultrapassagem pela principal)

regra (114)
Quando (condição de avançar na principal é permitida)
E (condição de avançar na secundária é permitida)
E (sentido dos trens é ímpar)
E (seção a frente é singela)
E (seção a frente permite avanço de trem ímpar)
Então (de preferência para ultrapassagem pela principal)

regra (115)

Quando (condição de avançar na principal é permitida)
E (condição de avançar na secundária é permitida)
E (sentido dos trens é ímpar)
E (seção a frente é dupla)
Então (de preferência para ultrapassagem pela principal)

regra (116)
Quando (condição de avançar na principal é permitida)
E (condição de avançar na secundária é permitida)
E (sentido dos trens é par)
E (seção a frente é dupla)
Então (de preferência para ultrapassagem pela principal)

regra (022)
Quando (condição de avançar na linha principal não é conhecida)
E (prioridade do trem1 é maior do que prioridade do trem2)
Então (determine condição do trem1 avançar na linha principal)

regra (023)
Quando (condição de avançar na linha principal não é conhecida)
E (prioridade do trem1 é menor do que prioridade do trem2)
Então (determine condição do trem2 avançar na linha principal)

regra (024)
Quando (condição de avançar na linha secundária não é conhecida)
E (prioridade do trem1 é maior do que prioridade do trem2)
Então (determine condição do trem1 avançar na linha secundária)

regra (025)
Quando (condição de avançar na linha secundária não é conhecida)
E (prioridade do trem1 é menor do que prioridade do trem2)
Então (determine condição do trem2 avançar na linha secundária)

- Ultrapassagem em Linha Singela (tipo 5):

módulo (ruleset 5):

abstração funcional (objetivo): determine preferência para trem encostar;

preferências:

oper1 - trem não prioritário aguarda no pátio anterior
oper2 - trem não prioritário aguarda no pátio posterior
oper3 - não há preferência
oper4 - rejeição
oper5 - retarda trem prioritário na principal;

importa (meta-regra): condição de trem encostar (parar);

produções:

regra (090)
Quando (condição de encostar no pátio anterior é obrigatória)
E (condição de encostar no pátio posterior é não permitida)
Então (de preferência para encostar trem na seção)

regra (091)
Quando (condição de encostar no pátio anterior é permitida)
E (condição de encostar no pátio posterior é não permitida)
Então (de preferência para encostar trem na seção)

regra (092)
Quando (condição de encostar no pátio anterior é não permitida)
E (condição de encostar no pátio posterior é obrigatória)
Então (de preferência para encostar trem na seção a frente)

regra (093)
Quando (condição de encostar no pátio anterior é não permitida)
E (condição de encostar no pátio posterior é permitida)
Então (de preferência para encostar trem na seção a frente)

regra (094)
Quando (condição de encostar no pátio anterior é permitida)
E (condição de encostar no pátio posterior é obrigatória)
Então (de preferência para encostar trem na seção a frente)

regra (095)
Quando (condição de encostar no pátio anterior é obrigatória)
E (condição de encostar no pátio posterior é permitida)
Então (de preferência para encostar trem na seção)

regra (096)
Quando (condição de encostar no pátio anterior é permitida)
E (condição de encostar no pátio posterior é permitida)
E (sentido dos trens é ímpar)
E (seção a frente é singela)
E (seção a frente permite avanço para trem ímpar)
Então (não há preferência para condição de encostar trem)

regra (097)
Quando (condição de encostar no pátio anterior é permitida)
E (condição de encostar no pátio posterior é permitida)
E (sentido dos trens é par)
E (seção a frente é singela)
E (seção a frente permite avanço para trem par)
Então (não há preferência para condição de encostar trem)

regra (098)
Quando (condição de encostar no pátio anterior é permitida)
E (condição de encostar no pátio posterior é permitida)
E (sentido dos trens é ímpar)
E (seção a frente é dupla)
Então (não há preferência para condição de encostar trem)

regra (099)
Quando (condição de encostar no pátio anterior é permitida)
E (condição de encostar no pátio posterior é permitida)
E (sentido dos trens é par)
E (seção a frente é dupla)
Então (não há preferência para condição de encostar trem)

regra (100)
Quando (condição de encostar no pátio anterior é não permitida)

E (condição de encostar no pátio posterior é não permitida)
Então (rejeição)

regra (101)
Quando (condição de encostar no pátio anterior é obrigatória)
E (condição de encostar no pátio posterior é obrigatória)
Então (não há preferência para condição de encostar trem)

regra (102)
Quando (condição de encostar no pátio anterior é não permitida)
E (sentido dos trens é ímpar)
E (seção a frente é singela)
E (seção a frente não permite avanço para trem impar)
Então (retarda ultrapassagem)

regra (103)
Quando (condição de encostar no pátio anterior é não permitida)
E (sentido dos trens é par)
E (seção a frente é singela)
E (seção a frente não permite avanço para trem par)
Então (retarda ultrapassagem)

regra (104)
Quando (condição de encostar no pátio anterior é permitida)
E (sentido dos trens é ímpar)
E (seção a frente é singela)
E (seção a frente não permite avanço para trem impar)
Então (de preferência para encostar trem na seção)

regra (105)
Quando (condição de encostar no pátio anterior é permitida)
E (sentido dos trens é par)
E (seção a frente é singela)
E (seção a frente não permite avanço para trem par)
Então (de preferência para encostar trem na seção)

regra (016)
Quando (condição de encostar no pátio anterior não é conhecida)
E (prioridade do trem1 é maior que prioridade do trem2)
Então (determine condição do trem2 encostar na seção)

regra (017)
Quando (condição de encostar no pátio anterior não é conhecida)
E (prioridade do trem2 é maior que prioridade do trem1)
Então (determine condição do trem1 encostar na seção)

regra (018)
Quando (condição de encostar no pátio posterior não é conhecida)
E (prioridade do trem1 é maior que prioridade do trem2)
E (sentido dos trens é ímpar)
Então (determine condição do trem2 encostar seção a frente sentido imp)

regra (019)
Quando (condição de encostar no pátio posterior não é conhecida)
E (prioridade do trem1 é maior que prioridade do trem2)
E (sentido dos trens é par)
Então (determine condição do trem2 encostar seção a frente sentido par)

regra (020)
Quando (condição de encostar no pátio posterior não é conhecida)
E (prioridade do trem2 é maior que prioridade do trem1)
E (sentido dos trens é ímpar)
Então (determine condição do trem1 encostar seção a frente sentido imp)

regra (021)
Quando (condição de encostar no pátio posterior não é conhecida)
E (prioridade do trem2 é maior que prioridade do trem1)
E (sentido dos trens é par)
Então (determine condição do trem1 encostar seção a frente sentido par)

5.4.4 - Nível de Restrições

Conforme pudemos verificar, no nível decisório, os módulos de regras responsáveis pela determinação de preferência segundo o tipo de conflito possuem um conjunto de meta-regras que invocam dois módulos responsáveis pela verificação das condições de algum trem parar, aguardar ou encostar (nos casos de conflitos em linha singela) e avançar ou ultrapassar (nos casos de conflito em linha dupla).

Estes módulos, considerados módulos de restrições,estão situados no nível inferior da hierarquia , e são responsáveis pela verificação das restrições físicas e operacionais do sistema, possuindo as seguintes características básicas.

- Condição de Trem Parar:

módulo (ruleset 3):
abstração funcional (objetivo):determine condição de trem parar;
preferências: permitida, não permitida e obrigatória;

produções:

regra (069)
Quando (sentido do trem é par)
E (parada do trem no pátio posterior é obrigatória)
Então (condição é obrigatória)

regra (070)
Quando (sentido do trem é ímpar)
E (parada do trem no pátio anterior é obrigatória)
Então (condição é obrigatória)

regra (071)
Quando (trem já aguarda em pátio de cruzamento)

Então (condição é obrigatória)

regra (072)
Quando (sentido do trem é par)
E (comprimento do trem é maior que pátio posterior)
Então (condição é não permitida)

regra (073)
Quando (sentido do trem é ímpar)
E (comprimento do trem é maior que pátio anterior)
Então (condição é não permitida)

regra (074)
Quando (sentido do trem é ímpar)
E (pátio anterior desativado)
Então (condição é não permitida)

regra (075)
Quando (sentido do trem é par)
E (pátio posterior desativado)
Então (condição é não permitida)

regra (076)
Quando (sentido do trem é ímpar)
E (não é possível demarragem no pátio anterior)
Então (condição é não permitida)

regra (077)
Quando (sentido do trem é par)
E (não é possível demarragem no pátio posterior)
Então (condição é não permitida)

regra (078)
Quando (sentido do trem é ímpar)
E (é possível demarragem no pátio anterior)
E (comprimento do trem é menor que o pátio anterior)
E (pátio anterior está desocupado)
E (trem não aguarda em cruzamento)
Então (condição é permitida)

regra (079)
Quando (sentido do trem é par)
E (é possível demarragem no pátio posterior)
E (comprimento do trem é menor que o pátio posterior)
E (pátio posterior está desocupado)
E (trem não aguarda em cruzamento)
Então (condição é permitida)

regra (080)
Quando (sentido do trem é ímpar)
E (pátio anterior está ocupado)
Então (condição é não permitida)

regra (081)
Quando (sentido do trem é par)
E (pátio posterior está ocupado)
Então (condição é não permitida)

- Condição de Trem Avançar:

módulo (ruleset 4):

abstração funcional (objetivo): determine condição de trem avançar;

preferências: permitida e não permitida;

produções:

regra (082)
Quando (sentido do trem é ímpar)
E (linha principal da seção permite avanço para trem ímpar)
Então (condição é permitida)

regra (083)
Quando (sentido do trem é ímpar)
E (linha principal da seção não permite avanço para trem ímpar)
Então (condição é não permitida)

regra (084)
Quando (sentido do trem é par)
E (linha principal da seção permite avanço para trem par)
Então (condição é permitida)

regra (085)
Quando (sentido do trem é par)
E (linha principal da seção não permite avanço para trem par)
Então (condição é não permitida)

regra (086)
Quando (sentido do trem é ímpar)
E (linha secundaria da seção permite avanço para trem ímpar)
Então (condição é permitida)

regra (087)
Quando (sentido do trem é ímpar)
E (linha secundária da seção não permite avanço para trem ímpar)
Então (condição é não permitida)

regra (088)
Quando (sentido do trem é par)
E (linha secundária da seção permite avanço para trem par)
Então (condição é permitida)

regra (089)
Quando (sentido do trem é par)
E (linha secundária da seção não permite avanço para trem par)
Então (condição é não permitida)

5.5 - Desenvolvimento de Regras Heurísticas

5.5.1 - Situações de Conflitos Clássicos

O sistema de produção desenvolvido para o SIGT, consiste de um protótipo inicial com o objetivo maior de garantir a viabilidade de implantação do sistema de controle e gerenciamento de trens. Desta forma, devido ao curto período de desenvolvimento, não foi possível a aquisição e representação de todo conhecimento relativo ao domínio do problema. Este conhecimento incluí, óbviamente, o conhecimento completo de heurísticas, essencial no processo de planejamento e "scheduling".

Entretanto, dada a nossa experiência de trabalho e vivência no meio ferroviário, foi possível estabelecer para o SIGT algumas regras de extrema importância, empregadas por controladores, que contém o conhecimento prático e especialista, utilizado no controle e gerenciamento de trens.

Em seu trabalho, B.Szpigel [1972] apresenta dois tipos clássicos de conflito entre dois trens em um trecho qualquer de uma malha ferroviária com linha singela e pátios de cruzamento:

- Conflito de Cruzamento:

Se os dois trens tem sentidos contrários, temos um conflito de cruzamento na forma da figura 39.

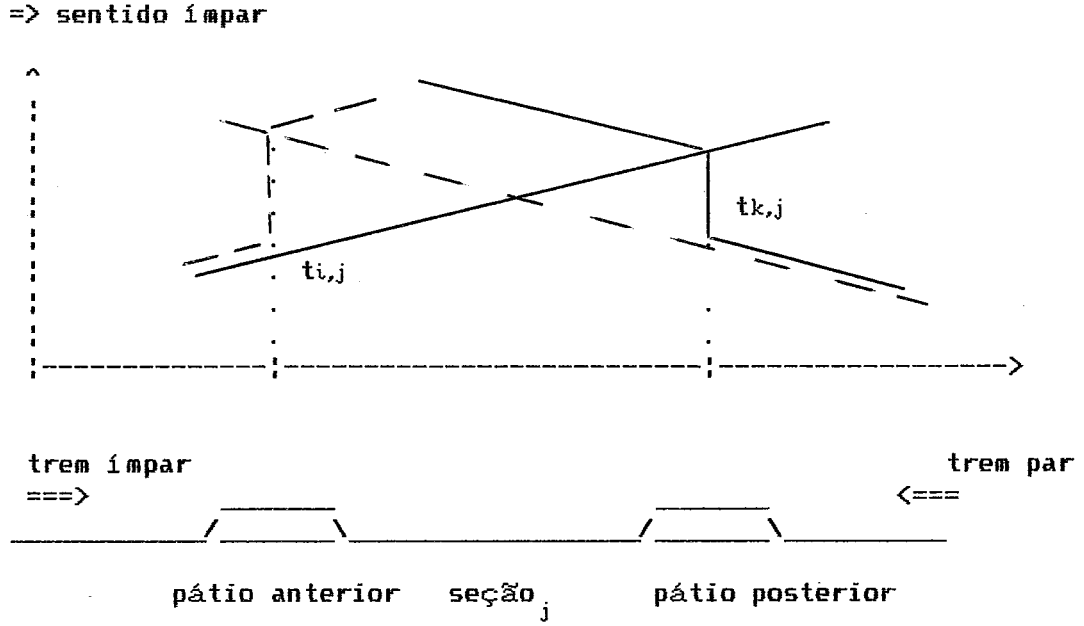


figura 39: conflito de cruzamento.

Sendo $t_{i,j}$ e $t_{k,j}$ os tempos de início de processamento (operação) dos trens ímpar e par na seção "j", $p_{i,j}$ e $p_{k,j}$ os tempos de processamento (percurso) dos trens na mesma seção, é possível identificarmos o conflito verificando as seguintes condições:

$$t_{i,j} + p_{i,j} \geq t_{k,j} \quad \text{se} \quad t_{k,j} \geq t_{i,j} \qquad (114)$$

e

$$t_{i,j} + p_{i,j} \geq t_{k,j} \quad \text{se} \quad t_{k,j} \geq t_{i,j}, \qquad (115)$$

admitindo a parada do trem ímpar no pátio anterior da seção "j" , ou a parada do trem par no pátio posterior.

- Manutenção do "Hedway":

Se os dois trens tem o mesmo sentido, temos um conflito de manutenção do "headway" na forma da figura 40.

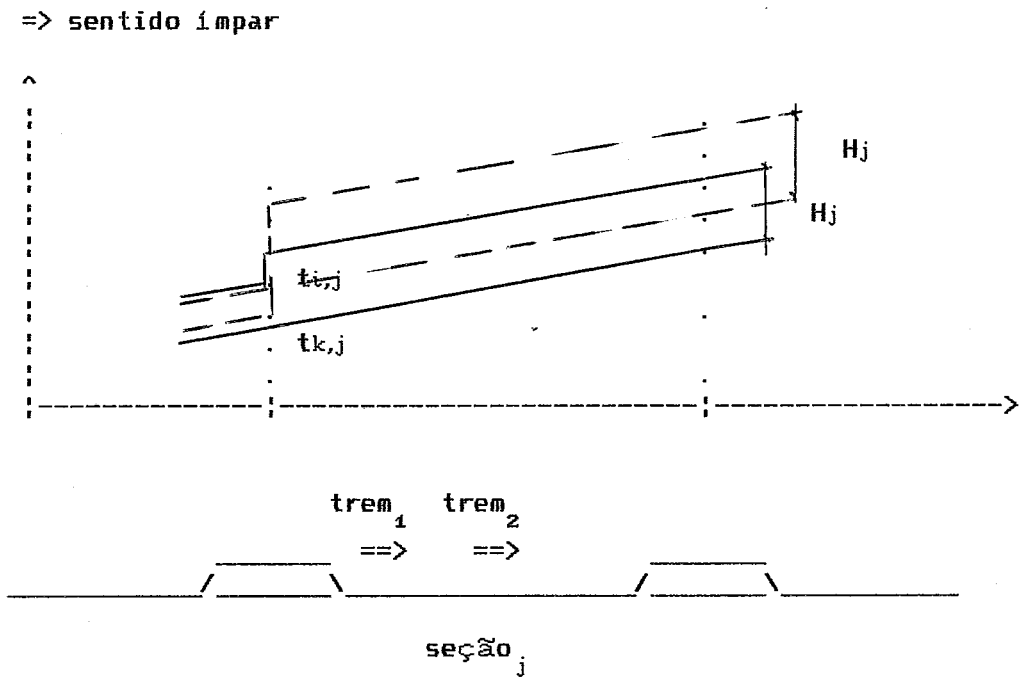


figura 40: manutenção do "headway".

Sendo $t_{i,j}$ e $t_{k,j}$ os tempos de início de processamento (operação) dos trens $trem_1$ e $trem_2$ na seção "j" ,e H_j o tempo máximo de "headway" entre dois trens no sentido ímpar na seção "j", identificamos o conflito , verificando as seguintes condições:

$$t_{k,j} + H_j \geq t_{i,j} \quad \text{se} \quad t_{i,j} \geq t_{k,j} \qquad (116)$$

e

$$t_{i,j} + H_j \geq t_{k,j} \quad \text{se} \quad t_{k,j} \geq t_{i,j}\,, \qquad (117)$$

admitindo a parada do $trem_1$ ou a parada do $trem_2$, no pátio anterior da seção "j".

- Conflito de Ultrapassagem:

Entretanto, ao longo do processo de planejamento, a identificação de conflitos e respectiva tomada de decisão não parece uma tarefa tão simples quanto exposto no trabalho de Szpigel.

Por exemplo, no trabalho de Szpigel não é considerado a possibilidade de conflito de ultrapassagem entre dois trens em uma seção qualquer da ferrovia. Neste caso, temos a situação da figura 41.

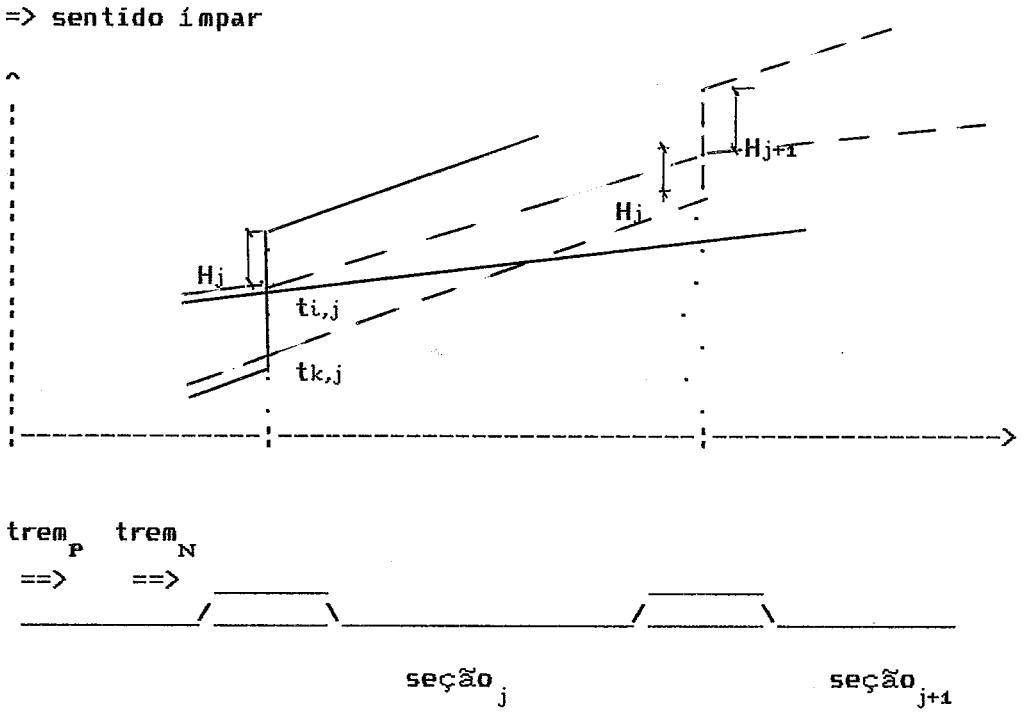


figura 41: conflito de ultrapassagem.

Sendo $t_{i,j}$ e $t_{k,j}$ os tempos de início de processamento (operação) dos trens prioritário e não prioritário na seção "j", $p_{i,j}$ e $p_{k,j}$ os tempos de percurso na mesma seção e H_j o tempo máximo de "headway" entre dois

trens no sentido ímpar, identificamos o conflito de ultrapassagem, verificando as seguintes condições:

$$t_{i,j} \geq t_{k,j} \qquad (118)$$

e

$$t_{k,j} + p_{k,j} >/ t_{i,j} + p_{i,j} - H_j \ , \qquad (119)$$

admitindo a parada do trem não prioritario no pátio anterior ou posterior da seção "j".

A identificação dos conflitos de cruzamento, manutenção do "headway" e ultrapassagem, bem como o instante no qual os conflitos aconteceriam , é feita, gráficamente, na grade de trens (diagrama espaço-tempo) através da projeção das curvas de velocidade dos trens. Denominaremos este processo de método projetivo.

Devemos destacar, também, que Szpigel direcionou seu modelo para a solução de conflitos em linha singela, não considerando a hipótese de solução de conflitos em linha dupla. Para tanto, determinamos nóvos métodos de decisão, os quais exigiram o desenvolvimento de nóvos procedimentos para a atualização da tabela de tempos de forma a garantir a viabilidade das restrições estruturais do problema de planejamento e "scheduling".

5.5.2 - Situações de Conflito Complexas

A complexidade do processo decisório levou-nos a um estudo detalhado de diversas situações reais de operação que permitiram o desenvolvimento de diversas heurísticas no intuito de facilitar e, em alguns casos, tornar viável, a elaboração do plano de horários para o gerenciamento do tráfego de trens.

Entre estas regras ou heurísticas desenvolvidas, podemos destacar as regras que verificam a inexistência de conflitos, regras que determinam a preferência de ondas de trens, regras que previnem a formação de bloqueios e as regras que verificam as restrições de capacidade ou ocupação dos pátios de cruzamento.

- Inexistência de Conflito:

=> sentido ímpar

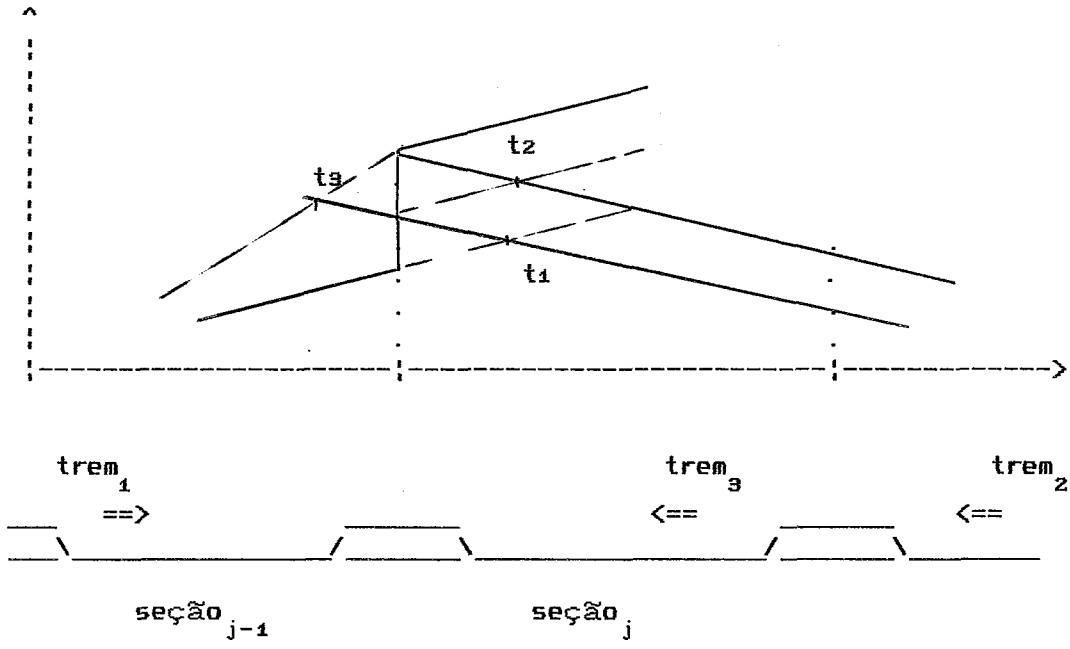


figura 42: inexistência de conflito.

Neste caso, ver figura 42, respeitando a ordem cronológica ,teremos, primeiramente, a ocorrência de um conflito de cruzamento entre os trens $trem_1$ e $trem_3$ (instante t1), na $seção_j$ da ferrovia. Supondo que a decisão seja parar o $trem_1$, teremos, em seguida, o conflito de cruzamento entre os trens $trem_1$ e $trem_2$ (instante t2), também, na $seção_j$ da ferrovia. Novamente, vamos considerar a parada do $trem_1$ (permanece parado). Entretanto, ao atualizarmos o valor do tempo de inicio de operação do $trem_1$ na $seção_j$, teremos, pelo método projetivo,a identificação de um falso conflito de cruzamento na $seção_{j-1}$ entre o $trem_1$ e o $trem_3$ (instante t3).

Para evitarmos a ocorrência deste falso conflito, introduzimos uma heurística que contém um predicado que verifica a existência de algum trem envolvido no conflito aguardando, ou seja:

```
regra (039)
Quando (trem2 já aguarda no pátio de cruzamento)
Ou (trem1 também já aguarda no pátio de cruzamento)
```

Então (não existe conflito de cruzamento)

O predicado lógico AguardaTrem (trem, seção) é exportado do módulo que implementa e opera sobre a tabela de tempos, juntamente, com o procedimento SetTremAguarda (trem, seção, valor lógico) , o qual possibilita a atualização, a priori, da condição de aguardar de todos os trens em todas as seções. Esta informação é controlada por uma matriz de valores lógicos que é atualizada, antes de invocarmos o sistema de produção , em função da decisão dos últimos conflitos antecessores, permitindo manter a consistência da base de conhecimento em relação a este tipo de informação decorrente de possíveis alterações provocadas pelo processo decisório ao longo do planejamento.

- Prevenção de Bloqueio:

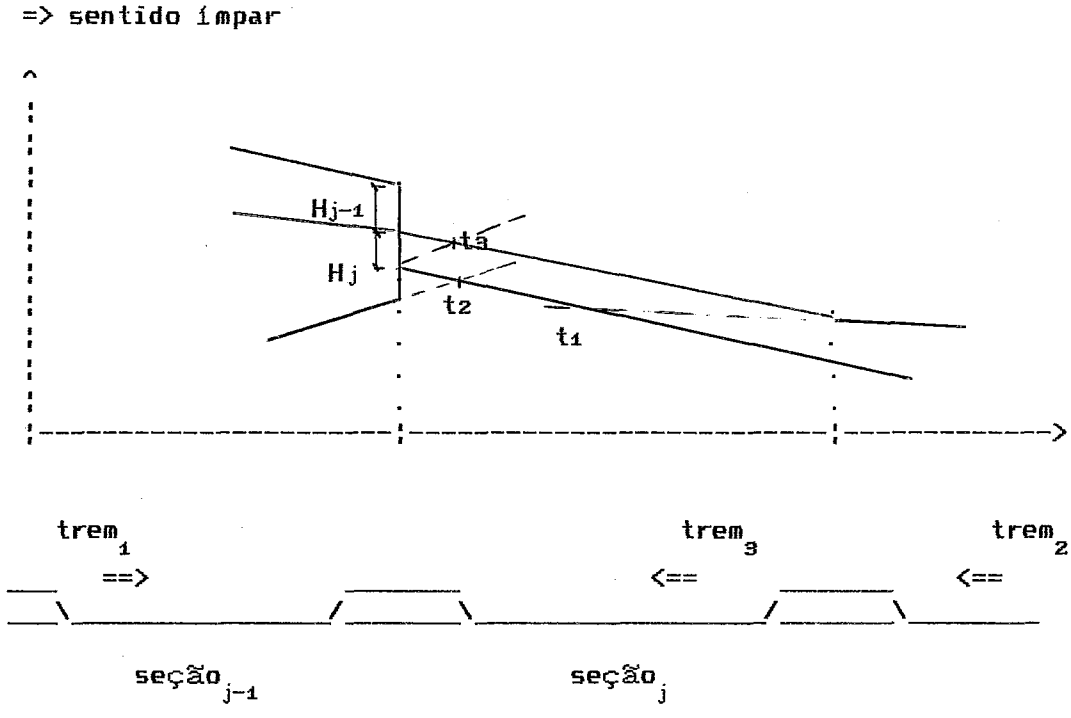


figura 43: prevenção de bloqueio com projeção de cruzamento na seção$_j$.

Vamos supor, que, na figura 43, ao decidirmos o primeiro conflito, relativo a ultrapassagem do trem$_3$ pelo trem$_2$ (instante t1), na seção$_j$, optassemos pela parada do trem$_3$ no pátio posterior (segundo o sentido par). Neste caso, teríamos a ocorrencia de um ímpasse, pois o segundo conflito, relativo ao cruzamento entre os trens

$trem_1$ e $trem_2$ (instante t2) na $seção_j$, não teria solução, já que a solução do conflito anterior determinou a ocupação do pátio de cruzamento pelo $trem_3$, e permitiu o avanço do $trem_2$.

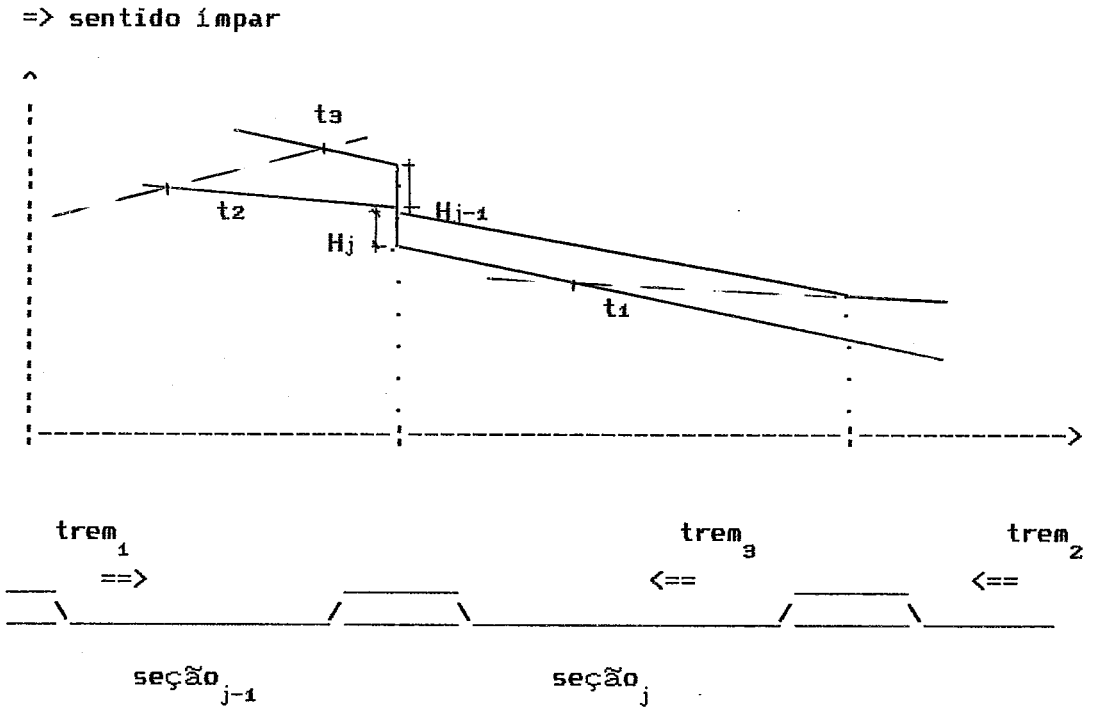


figura 44: prevenção de bloqueio com projeção de cruzamento na $seção_{j-1}$.

Entretanto, se a projeção do conflito de cruzamento entre o $trem_1$ e o $trem_2$, ocorresse na $seção_{j-1}$, conforme a situação da figura 44, teríamos a ocupação do pátio de cruzamento pelo $trem_3$, impedindo a parada do $trem_2$, determinando a opção de pararmos o $trem_1$ no pátio anterior da $seção_{j-1}$. Caso no instante decisório do primeiro conflito (conflito de ultrapassagem) o $trem_1$ já tivesse avançado na $seção_{j-1}$, a parada do $trem_1$ no segundo conflito não seria possível, tornando, portanto, obrigatória, a parada do $trem_3$ no pátio anterior (segundo o sentido par) da $seção_j$.

Neste caso, se o $trem_1$ não tivesse avançado anteriormente, poderiamos optar pela parada do $trem_1$ na solução do conflito de cruzamento como veremos adiante.

A preferência para encostarmos o $trem_3$ no pátio anterior (segundo o sentido par), é expressa pela regra:

regra (105)
Quando (condição de encostar no pátio anterior é permitida)
E (sentido dos trens é par)
E (seção a frente é singela)
E (seção a frente não permite avanço para trem par)
Então (de preferência para encostar trem na seção)

Porém, caso a condição de parada do $trem_3$ no pátio anterior (segundo o sentido par) não fosse permitida, teríamos que retardar a ultrapassagem ocasionando um atraso do $trem_2$. Esta situação é expressa pela regra:

regra (103)
Quando (condição de encostar no pátio anterior é não permitida)
E (sentido dos trens é par)
E (seção a frente e éingela)
E (seção a frente não permite avanço para trem par)
Então (retarda ultrapassagem)

- Cruzamento com Onda de Trens:

A segunda opção decisória (parada do $trem_1$ na situação anterior), levou-nos ao desenvolvimento de uma heurística que determina a preferência de parada para um trem quando ocorre um provável cruzamento com uma onda de trens que se forma em sentido contrário, segundo a situação da figura 45. Neste caso, a onda de trens fica caracterizada pela ocorrência de conflitos simultâneos na mesma seção.

No exemplo anterior , podemos imaginar a ocorrência do primeiro conflito, relativo ao cruzamento entre os trems $trem_1$ e $trem_3$ (instante t_1) na $seção_j$, e, em seguida, o cruzamento dos trens $trem_1$ e $trem_2$ (instante t_2) na mesma seção, resultante da parada do $trem_1$ no conflito anterior. Neste caso, a heurística determina a solução do primeiro conflito (obriga a parada do $trem_1$). Para a segunda decisão , relativa ao cruzamento entre o $trem_1$ e $trem_2$, não é estabelecido, a primeira vista, algum tipo de preferência. É interessante observarmos que, a utilização desta heurística, previne a ocorrência de conflitos inexistentes.

Convém ressaltarmos, também, que a ocorrência de uma onda de trens , pode, em alguns casos, estar acompanhada de um conflito de manutenção do "headway" ou de ultrapassagem (caso anterior). Neste

sentido, a adoção desta heurística ou da heurística anterior dependerá do primeiro tipo de conflito.

Ou seja, caso o primeiro conflito seja entre os trens de mesmo sentido (trens que formam a onda) o sistema adotará a primeira heurística (parada de trem no pátio anterior). Entretanto, caso o primeiro conflito seja de cruzamento, o sistema determinará preferência para parar o trem em sentido contrário a onda de trens.

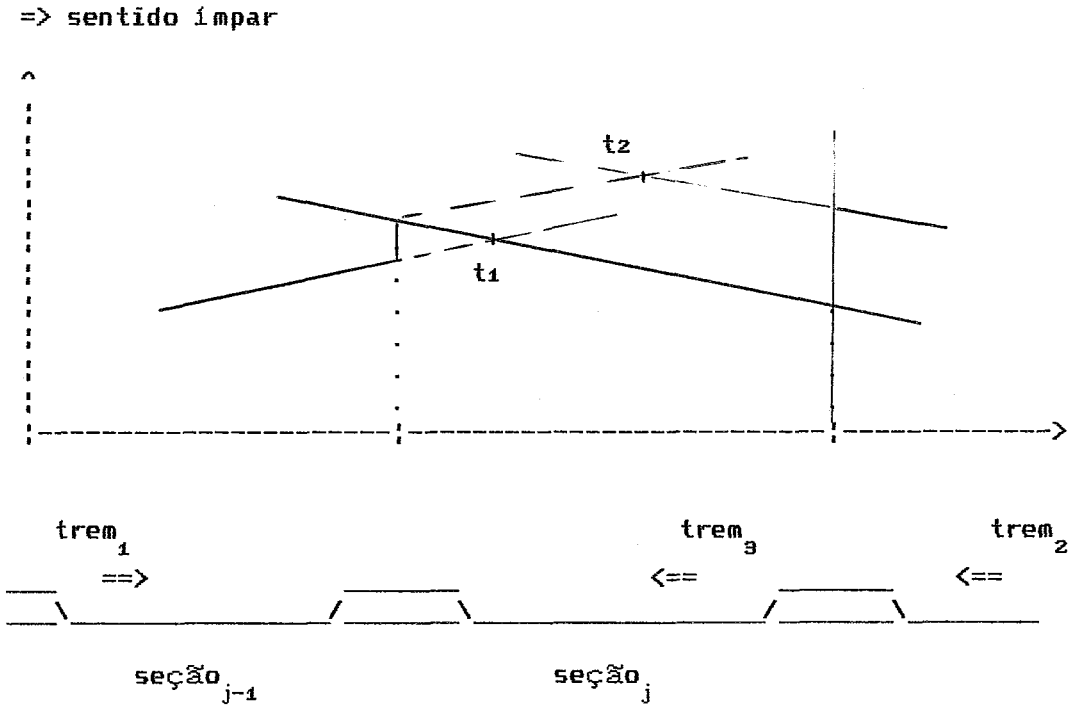


figura 45: cruzamento com onda de trens.

A verificação de uma onda de trens no conflito de cruzamentos, bem como a verificação da ocorrência de bloqueio em conflitos de ultrapassagem é considerada antes de invocarmos o sistema de produção.

Para tanto, utilizamos uma teoria elementar de tempo e ação (J.F. Allen [1984]), que será descrita posteriormente, incluíndo alguns predicados para o cálculo de intervalos e informações da tabela de tempos, relativas ao posicionamento atualizado dos trens. Quando detectamos uma onda de trens na seção$_j$, tornamos a condição de "status" do pátio posterior (segundo a convenção) da seção$_j$ ocupado, tornando, desta forma, obrigatória a parada do trem$_1$ no pátio anterior da seção$_j$. Este procedimento é traduzido nas seguintes

regras:

regra (081)
Quando (sentido do trem é par)
E (pátio posterior está ocupado)
Então (condição é não permitida)

regra (041)
Quando (condição de parar trem1 é permitida)
E (condição de parar trem2 é não permitida)
Então (de preferência para parar trem1)

- Manutenção do "Headway" com Conflito de Cruzamento:

Uma outra forma de heurística, que também verifica a condição de trens aguardando, se refere a decisão de manutenção de "headway" entre dois trens após a resolução de um conflito de cruzamento com um terceiro trem. Esta situação , muito comun no processo decisório, fica representada na forma da figura 46.

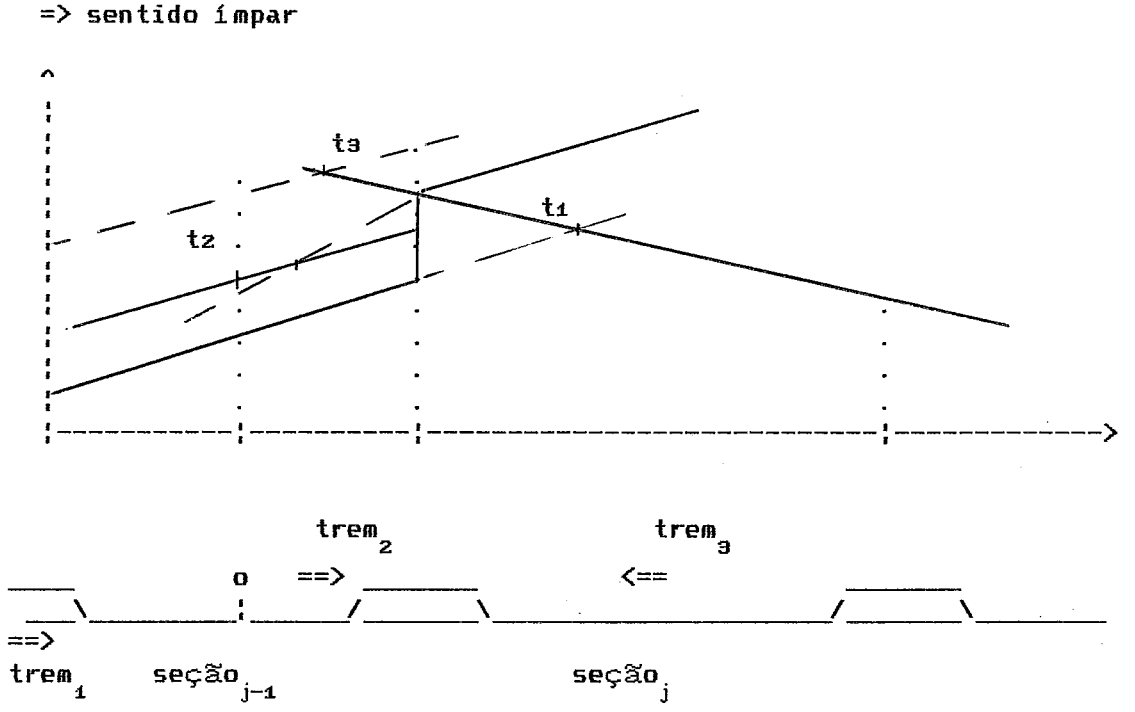


figura 46: manutenção do "headway" com conflito de cruzamento.

Neste exemplo, podemos observar que não haveria, anteriormente, conflito de manutenção do "headway" entre os trens $trem_1$ e $trem_2$. Entretanto, a decisão de parar o $trem_2$ no pátio anterior da $seção_j$ (

com a condição de que não tenha havido a formação de onda de trens) , referente ao conflito de cruzamento entre os trens $trem_2$ e $trem_3$ (instante t_1) na mesma seção, provocará a ocorrência do conflito de manutenção do "headway" (supondo que $trem_1$ e $trem_2$ tenham a mesma prioridade) na $seção_{j-1}$, pelo método projetivo, ao atualizarmos o início da operação do $trem_2$ na $seção_j$.

Este novo conflito, apesar de ser dependente lógicamente da decisão do primeiro conflito, pode, as vezes, ser cronológicamente inferior ao primeiro. Em condições operacionais, este conflito terá ocorrência, assim que o tempo de "headway" entre os trens $trem_1$ e $trem_2$ na $seção_{j-1}$ for superado. Entretanto, como o $trem_2$ esta aguardando no pátio, este instante se dará assim que o $trem_1$ passar pelo sinal automático da $seção_{j-1}$ (instante t2) que garante o cumprimento do tempo de "headway" ate o pátio posterior.

Ao analisarmos a hierarquia de conflitos, quanto a ocorrência do conflito de manutenção do "headway" entre os trens $trem_1$ e $trem_2$, segundo a solução do primeiro conflito, teremos as seguintes situações:

primeira: Caso ocorra o cruzamento do $trem_2$ com o $trem_3$ em cima do pátio de cruzamento (cruzamento em movimento), não teremos a parada do $trem_2$ e, consequentemente, não teremos conflito de manutenção do "headway";

segunda: Caso ocorra o cruzamento do $trem_2$ com o $trem_3$ em um instante inferior ou igual ao tempo de entrada do $trem_1$ na $seção_j$ menos o tempo de "headway" da $seção_{j-1}$ o conflito de manutenção do "headway" será cronológicamente superior ou igual ao primeiro conflito;

terceira: Caso ocorra o cruzamento do $trem_2$ com o $trem_3$ em um instante superior ao tempo de entrada do $trem_1$ na $seção_j$ menos o tempo de "headway" da $seção_{j-1}$, o conflito de manutenção do "headway" será cronológicamente inferior ao primeiro conflito, segundo a situação da figura 46.

Na solução do conflito de manutenção do "headway" temos, a primeira vista, quatro possibilidades de decisão:

primeira: parar o trem1 no pátio anterior da $seção_{j-1}$ o tempo necessário a não ocorrência do conflito de manutenção do

"headway";

segunda: manter o $trem_2$ no pátio posterior da $seção_{j-1}$;
terceira: parar o $trem_2$ no pátio anterior da $seção_{j-1}$, e
quarta: parar o $trem_1$ no pátio posterior da $seção_{j-1}$.

A terceira possibilidade, deve ser rejeitada, pois afeta a lógica da decisão anterior que determinou a parada do $trem_2$ no pátio posterior da $seção_{j-1}$. A quarta possibilidade, também deve ser descartada, já que provoca a ocorrência de um bloqueio, quando considerarmos o avanço do $trem_3$ na $seção_j$.

A primeira possibilidade é viável, no sentido de provocar, posteriormente, a ocorrência de um cruzamento normal entre os trens $trem_1$ e $trem_3$ (instante t_3), na $seção_{j-1}$ após a liberação do pátio de cruzamento pelo $trem_2$, ver figura 46. Esta decisão é factível, se os trens $trem_1$ e $trem_2$ tiverem um afastamento razoável, não sendo detectada a onda de trem (neste caso , a heurística coincide com a heurística de prevenção de bloqueios).

A segunda possibilidade provoca, posteriormente, a ocorrência do cruzamento entre os trens $trem_1$ e $trem_3$ na $seção_j$ (instante t_3), conforme a situação da figura 47. Neste caso, como não podemos parar o $trem_1$, pois o $trem_2$ aguarda a manutenção de "headway" , teremos, necessariamente, que parar o $trem_3$ no pátio anterior da $seção_j$. Entretanto, a parada do $trem_3$ afeta a lógica da decisão anterior que determina o avanço do mesmo. Portanto, esta possibilidade também deve ser rejeitada.

Caso a segunda possibilidade provoque a ocorrência do cruzamento na $seção_{j-1}$, refletindo a situação descrita na figura 47, teremos uma interpretação semelhante a terceira possibilidade. Ou seja, teremos como decisão viável a solução deste conflito entre os trens $trem_1$ e $trem_3$ (instante t_3), antes da partida do $trem_2$.

Neste caso, como detectamos, inicialmente, o conflito de manutenção de "headway" entre os trens $trem_1$ e $trem_2$ (instante t_2), teremos, necessariamente,que parar o $trem_1$ no pátio anterior da $seção_{j-1}$, já que o $trem_2$ aguarda no pátio posterior da mesma seção, tornando proibitiva a parada do $trem_3$ no mesmo pátio. Entretanto, a parada do $trem_1$, relativa a decisão do cruzamento com o $trem_3$, torna

desnecessária a manutenção do "headway" do $trem_2$, levando, novamente , a uma situação indesejável.

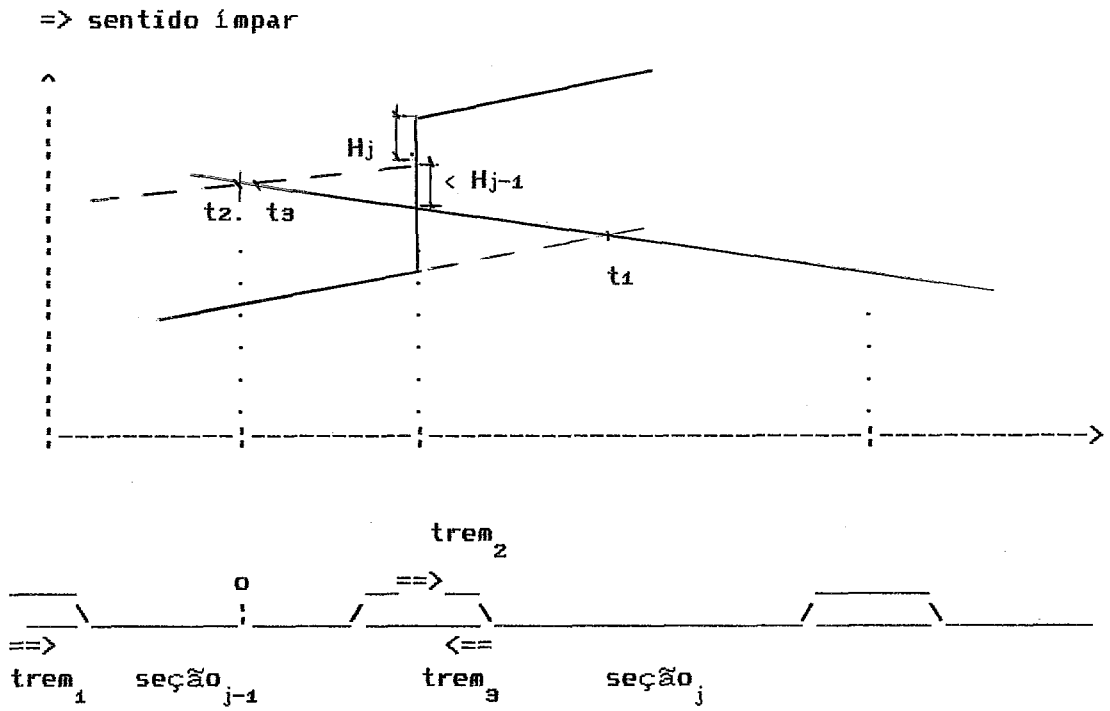


figura 47: manutenção do "headway", mantendo o $trem_2$ no pátio posterior da $seção_{j-1}$.

Resumindo, quando o sistema reconhece este tipo de conflito, a heurística determina a parada do $trem_1$ no pátio anterior. Caso contrário, teríamos caracterizada a formação de uma onda de trens, cuja heurística determinaria a parada do $trem_3$ no pátio anterior da $seção_j$, não provocando este tipo de situação.

- Inexistência de Conflito de Ultrapassagem:

Podemos ter, também, um caso particular de inexistência de ultrapassagem (instante t2),quando o $trem_1$ inicia a operação na $seção_j$ em um instante inferior ao futuro instante de saída do $trem_2$ menos o "headway" inicial da $seção_{j-1}$, conforme a descrição feita na figura 48.

Neste caso, teremos, posteriormente, a ocorrência do conflito de cruzamento entre os trens $trem_1$ e $trem_3$ (instante t3) na $seção_j$. Entretanto, devido a inexistência do conflito anterior de manutenção do "headway" entre os trens $trem_1$ e $trem_2$, forçamos o avanço do $trem_1$ na

seção$_j$, obrigando, a parada do trem$_3$.

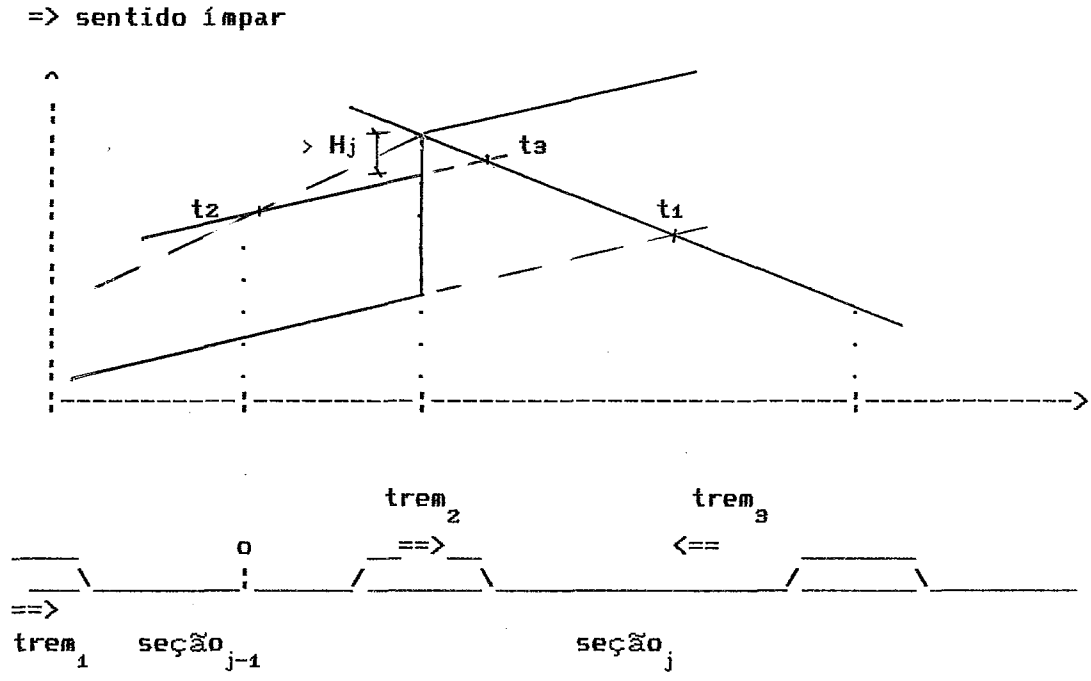


figura 48: inexistência de conflito de ultrapassagem.

Mas, como a parada do trem$_3$ afeta a lógica da decisão de cruzamento entre os trens trem$_2$ e trem$_3$ ocorrida anteriormente, teremos, a princípio, uma situação de bloqueio. Porém, na ocorrência deste tipo de situação, teíiamos, inicialmente, na solução do conflito de cruzamento entre os trens trem$_2$ e trem$_3$, a ocorrência de uma onda de trens , forçando a parada do trem$_3$, o que resolveria o problema.

5.5.3 - Situações Envolvendo Restrições

- Análise de Capacidade:

A verificação da ocupação das seções e, consequentemente, a atualização de seu "status", é feita antes de invocarmos o sistema de produção de modo a garantir a consistência destas informações na base de conhecimento ao longo do processso de elaboração de planos. A situação de conflito de capacidade foi amplamente demonstrada nas situações anteriores, sempre que a ocupação temporária de um pátio de cruzamento ou de uma linha, impedia a parada ou avanço de algum trem.

A verificação da condição de ocupação das seções é possível

através da utilização de informações dos conflitos realizados mais recentemente , que são armazenadas na estrutura representativa de estados. Estas estruturas, ver seção 3.6.6, além de representarem , como já vimos, a tabela de tempos e valores das funções de pertinência, contém informações a respeito dos conflitos já decididos, bem como informações necessárias a atualização da base de conhecimento , relacionadas a ocorrência dos últimos conflitos.

No nosso sistema, achamos necessário guardar as informações relativas a ocorrência, sómente, do conflito antecessor, que determina e caracteriza uma nova situação. Entre estas informações , destacamos: número da seção ocupada , código do trem que ocupou a seção , intervalo de ocupação incluindo o tempo de entrada e o tempo de saída e um código que estabelece o tipo de bloqueio.

Como exemplo, vamos imaginar que um trem ($trem_2$) em sentido ímpar, ocupe o pátio anterior da $seção_j$, relativo a decisão do cruzamento com o $trem_3$ e $trem_4$ (instante t_1 e t_2). Neste caso, existe um bloqueio do pátio para o trem em sentido par ($trem_4$, que avança na $seção_j$) e um bloqueio para o trem em sentido ímpar ($trem_1$, que avança na $seção_{j-1}$), obrigando a solução do conflito de cruzamento entre os trens $trem_1$ e $trem_4$ (instante t_4) na $seção_{j-1}$, através da parada obrigatória do $trem_1$ no pátio anterior da mesma seção, ver figura 49.

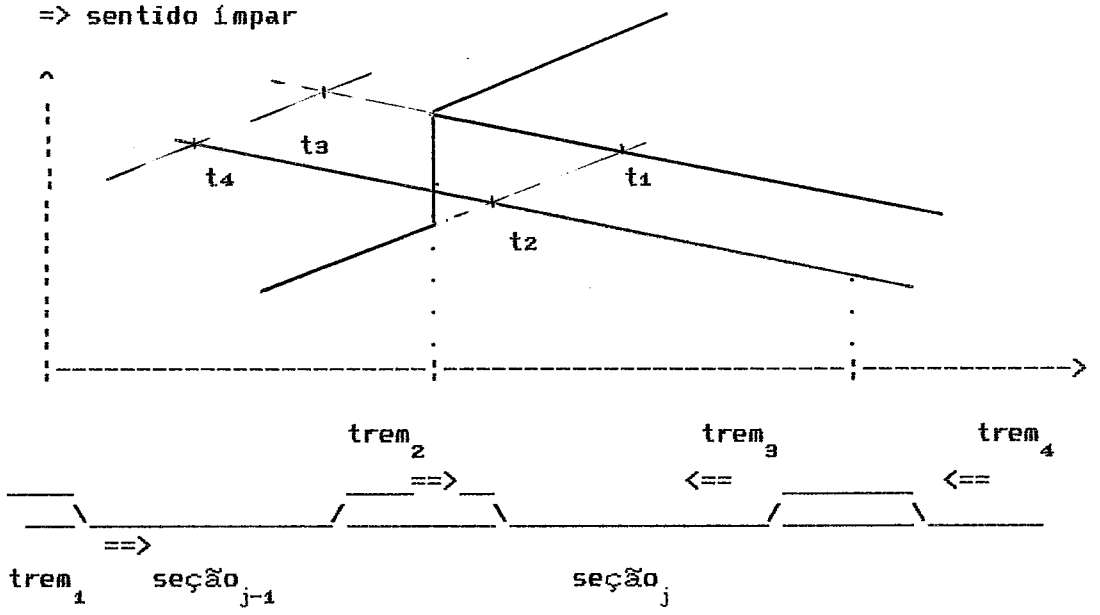


figura 49: análise de capacidade.

As informações representadas na estrutura estado, são atualizadas de forma que, a cada novo estado gerado, é incluido o efeito da decisão relativa a solução do conflito que caracterizava a situação do estado antecessor.

É importante ressaltarmos que, quando retornamos do sistema de produção, restauramos o status das seções e de todas as informações dinâmicas da base de conhecimento, incluíndo as informações relativas as condições de trem aguardando, já que as mudanças verificadas são dependentes de uma hierarquia de decisões particulares, associadas a um estado do espaço do problema. Estas mudancas, são propagadas ao longo do processo de planejamento , durante o processo de geração de estados, de estado pai para estado filho ou estado gerado. Neste sentido, podemos dizer que as informações dinâmicas da base de conhecimento espelham , sempre, a situação particular de uma determinada linha de ação, preservando , desta forma, a consistência do conhecimento.

- Restrições Físicas e Operacionais:

Finalmente, queremos retratar algumas restrições físicas e operacionais, traduzidas em regras, que são de fundamental importância para a preservação da viabilidade dos planos de horários e para a determinação de preferências.

Um exemplo típico de restrição operacional é quando, na existência de um conflito, verificamos, coincidentemente, a obrigatoriedade de parada de algum trem . Neste caso, determinamos a preferência para o mesmo parar, através das seguintes regras:

```
regra (069)
Quando (sentido do trem é par)
E (parada do trem no pátio posterior é obrigatória)
Então (condição é obrigatória)

regra (044)
Quando (condição de parar trem2 é obrigatória)
E (condição de parar trem1 é permitida)
Então (de preferência para parar trem2)
```

Para verificarmos, se a condição de algum trem par parar em um pátio de cruzamento, é permitida, teremos, a princípio, que respeitar as seguintes restrições físicas incorporadas a regra abaixo:

regra (079)
Quando (sentido do trem é par)
E (é possível demarragem no pátio posterior)
E (comprimento do trem é menor que o pátio posterior)
E (pátio posterior está desocupado)
E (trem não aguarda em cruzamento)
Então (condição é permitida)

Caso contrário, ou seja, caso alguma das condições da regra 79 não fossem atendidas , a condição de parada do trem par não seria , obviamente, permitida.

Capítulo VI

Técnicas Aplicadas ao Desenvolvimento do Sistema como uma Arquitetura Inteligente

Neste capítulo, será descrita a integração de sistemas baseados em conhecimento com as hipóteses fundamentais de Inteligência Artificial. Esta integração permitiu o desenvolvimento de arquiteturas inteligentes, dentre as quais, inserimos nosso sistema de controle e gerenciamento de trens. Abordaremos, também, as principais questões que delineam o desenvolvimento de sistemas inteligentes, como o uso combinado de técnicas de Inteligência Artificial e Pesquisa Operacional, e o desenvolvimento de modelos de simulação associados a sistemas especialistas.

6.1 - Arquiteturas Inteligentes

6.1.1 - Arquitetura SOAR

SOAR (J.E. Laird, A. Newell e P.S. Rosenbloom [1987]) é exemplo do desenvolvimento de uma arquitetura de conhecimento, considerada inteligente para a solução de problemas. Para tanto, SOAR emprega uma conduta inteligente, obedecendo a um conjunto de hipóteses ou fundamentos, que incluem:

1 - Hipótese do Sistema Físico Simbólico : como vimos anteriormente, esta é a hipótese básica para o estabelecimento de um sistema inteligente;

2 - Hipótese da Conduta Objetiva: esta hipótese determina o controle e emprego do conhecimento voltado para a satisfação de objetivos de forma racional;

3 - Hipótese do Espaço do Problema:no sentido de tornar operavél o sistema físico simbólico;

4 - Hipótese do Sistema de Produção: como forma de representar e organizar o conhecimento de longo prazo, dependente do domínio;

5 - Hipótese da Geração Automática de Sub-Objetivos: como forma de resolver impasses e diminuir o esforço de busca;

6 - Hipótese do Controle do Conhecimento: no sentido de afirmar

que alguma decisão pode ser controlada por alguma forma de conhecimento dependente ou independente do domínio;

7 - Hipótese dos Métodos Fracos: esta hipótese ressalta que os métodos de busca independentes do domínio constituem os métodos básicos de inteligência;

8 - Hipótese da Representação Elementar e Uniforme do Conhecimento: esta hipótese sugere uma forma de representação única e simples para o conhecimento declarativo e suas relações, e, finalmente,

9 - Hipótese de Aprendizado Uniforme: esta hipótese garante o acumulo automático de conhecimento (aprendizado) baseado na teoria "chunking" .

A idéia central de arquiteturas inteligentes, como SOAR, é o desenvolvimento , progressivo, de métodos de conhecimento intensivo, no sentido de realizarem menor esforço possível na tarefa de solução de problemas, atendendo aos objetivos estabelecidos.

6.1.2 - Processo Decisório (SOAR)

Para realizar uma tarefa, a arquitetura SOAR estabelece uma busca no espaço de estados do problema no sentido de satisfazer certos objetivos. Para tanto é necessário uma estrutura de representação ou estado e um conjunto de operadores que provocam a transformação de estados . O processo de busca consiste na sucessiva transformação de estados, a partir de um estado inicial, até atingir-se o estado solução que caracteriza a realização da tarefa.

Durante o processo de transformação, a arquitetura SOAR não estabelece um grafo de estados, a exemplo dos métodos de busca tradicionais. Neste sentido, a aplicação de um operador em um estado só é permitida quando se estabelece uma preferência para este operador. Ou seja, se existe conhecimento suficiente para a escolha de um operador, o processo de geração progride, caso contrário, temos estabelecida uma situação de ímpasse.

Na arquitetura SOAR, uma situação de ímpasse poderá ocorrer nos seguintes casos:

. existe um conjunto de operadores aplicáveis, mas não há informação suficiente para expressar preferência em um operador;

. existe um conjunto de operadores aplicáveis, mas conflitantes em relação aos objetivos;

. o operador existente não provoca mudanças no estado;

. existe rejeição, no sentido de não haver escolha viável de um operador.

Ao deparar-se com uma situação de impasse, SOAR determina a geração de um sub-objetivo que está diretamente associado a solução do mesmo. A realização deste sub-objetivo determinará a criação de uma preferência para um operador (casos 1 e 2), a rejeição do operador caso 3), ou a geração de um estado compatível com a aplicação do operador (caso 4). Como a cada sub-objetivo está associado um espaço de problema, SOAR determina a criação de uma hierarquia de espaços e objetivos.

Em SOAR, o espaço do problema (objetivos, estados, operadores, etc.) é implementado como um sistema de produção, que admite a execução paralela de regras.

Neste sistema, os operadores (procedimentos de controle) associados as situações onde os mesmos se aplicam , são representados por produções. Os procedimentos necessários a definição de um estado inicial, verificação do estado final, bem como aqueles necessários a criação de novos estados, também são representados por produções.

A memória de trabalho deste sistema, é constituida dos seguintes elementos:

. pilha de contextos: que contém elementos associados a um objetivo, espaço do problema, estado e operador, os quais definem os "slots" de um contexto;

. objetos: que podem representar o espaço do problema, objetivo, operador ou estado e

. preferências: que contém um objeto, o papel definido pelo mesmo (operador, estado, etc.), um contexto no qual a preferência se aplica e uma referência que determina o grau de aceitabilidade do objeto.

O procedimento de decisão, ou o processo de progressão (modificação de contextos atraves da alteração de um "slot"), compreende as seguintes fases:

1 - Fase de elaboração: nesta fase novos objetos, preferências e triplas O-A-V são inseridos na memória de trabalho;

2 - Fase de decisão: nesta fase, realiza-se o exame das preferências e da pilha de contextos. Neste caso, não ocorrendo ímpasse, como exemplo : existe preferência para a aplicação de um operador em um estado associado a um contexto (elemento da pilha de contextos), modifica-se o "slot" associado através da aplicação do operador e consequente recolocação do estado. No caso da existência de ímpasse, é determinada a geração de um sub-objetivo associado a um novo espaço de problema.

Na arquitetura SOAR, podemos afirmar que as funções de controle de busca são as preferências estabelecidas para os operadores que determinam a geração de novos estados, bem como a preferência estabelecida para a seleção de estados.

6.1.3 - Arquitetura SIGT

A arquitetura de nosso sistema, difere um pouco da arquitetura SOAR, embora ambas sejam movidas pela mesma idéia central.

Na arquitetura SOAR podemos dizer que:

1 - O método de solução de problemas está em segundo plano;

2 - O emprego de conhecimento intensivo, ou a realização do progresso sem busca, está em primeiro plano.

Neste sentido, a solução de problemas associados ao alcançe de sub-objetivos é necessária no sentido de estabelecer informações adicionais , ou a determinação de preferências na solução de ímpasses, permitindo o progresso.

Na arquitetura SIGT podemos dizer que:

1 - O método de solução do problema associado a operacionalização do sistema físico símbolico, e a realização de uma

busca heurística no espaço de estados do problema está em primeiro plano;

2 - O uso de conhecimento intensivo, representado em produções, é necessário para a determinação de preferência em operadores, ou resolução de ímpasses, no sentido de restringir o espaço de estados do problema, durante o processo de busca.

Portanto, na arquitetura SIGT, temos no primeiro plano a realização de busca, ao contrário da arquitetura SOAR. Esta busca será reduzida quanto maior for o conhecimento intensivo representado no sistema de produção . Já, na arquitetura SOAR, o processo de busca está associado a ocorrência de ímpasses, desta forma, se não houver no primeiro plano a ocorrência de ímpasse, não haverá, consequentemente , realização de busca.

6.1.4 - Processo Decisório (SIGT)

Quando utilizamos um processo de busca no espaço de estados do problema, associado a algum método fraco de Inteligência Artificial, o processo de seleção de estados e operadores é feito de uma forma mais simples e eficiente.

Este processo utiliza, normalmente, uma lista de estados abertos para representar a pilha de contextos, sendo que a seleção de estados nesta lista e decorrente do tipo de método utilizado ("best-first", A*, IDA* , etc.). A escolha de operadores é decorrente do tipo de situação associado ao estado selecionado e a estrutura do problema.

No nosso sistema, na lista de abertos, cada estado contém informações relativas a:

- avaliação ou satisfação de objetivos;
- preferências;
- hierarquia de decisões;
- estado (tabela de tempos);
- informações temporais (decorrente da solução de conflito antecessores).

Na arquitetura SIGT, incorporamos ao sistema de produção uma

quantidade de conhecimento suficiente , desenvolvendo heurísticas com capacidade de reconhecer situações de ímpasse e escolher a forma de solução apropriada determinando a preferência em operadores. Quando invocamos o sistema de produção, determinamos a geração de sub-objetivos associados a um espaço de problema, constituído de um objetivo, produções, objetos da memória de trabalho e de informações da base de conhecimento (métodos de "frames") dada a concepção hierárquica deste sistema.

A exemplo da arquitetura SOAR, rejeitamos operadores que levam a situações indesejáveis. Entretanto, para operadores com graus de satisfação aceitáveis, determinamos a aplicação paralela no estado selecionado, bem como o cálculo das funções de avaliação.

Neste sentido, utilizando um algoritmo do tipo "best-first", obtemos uma semântica de preferências muito superior a estabelecida na arquitetura SOAR (lembrando que a cada operador associamos uma função de avaliação que assume valores no intervalo fechado [0.0..1.0]). Esta função expressa a idéia de dominância "fuzzy" associada aos objetivos do problema, sendo uma forma natural de exprimir a qualidade dos operadores em relação aos objetivos estabelecidos.

Conforme visto anteriormente, a seleção de estados na arquitetura SIGT, de modo a permitir o progresso da busca , está associada a utilização de um método heurístico de busca que efetuará o controle do conhecimento no espaço de estados do problema. Entre os métodos utilizados pelo SIGT podemos destacar:

- Algoritmo A*:

Neste caso, o processo decisório fica restrito aos seguintes passos principais:

1 - Seleção do estado com melhor preferência (situado no topo da lista) entre os estados abertos (contextos);

2 - Seleção de operadores segundo o contexto escolhido (análise de conflitos). Caso o estado escolhido represente o estado final (ausência de conflitos) termine;

3 - Determinação de preferências no conjunto de operadores aplicáveis em função do conhecimento intensivo (restrições , heurísticas, etc);

4 - Avaliação dos operadores (semântica de preferências) segundo os objetivos do problema e eliminação dos operadores segundo os critérios de poda;

5 - Geração paralela de estados sucessores segundo a aplicação dos operadores remanescentes. Em caso de falha retorne ao passo 1;

6 - Inserção dos estados gerados no espaço do problema (lista de abertos) em função do valor de avaliação, e destruição do estado anterior;

7 - Retorno ao passo 1.

O processo decisório termina com uma solução ótima devido a admissibilidade da heurística associada a função de avaliação.

A determinação de preferências em função do conhecimento intensivo, expressa a solução de sub-problemas específicos, que levariam a situações de ímpasse (rejeição). Vale ressaltar que, neste caso, o sistema SIGT estabelece uma semântica exclusiva de preferências , ou seja , verificamos a aceitabilidade ou a rejeitabilidade dos operadores. A semântica de preferências em relação a desejabilidade no sentido de um operador ser melhor ou pior do que o outro é feita pelo passo 4 do algoritmo.

Quando interrompemos a execução do algoritmo, temos em mão uma solução parcialmente viável em relação ao nível de profundidade e parcialmente ótima em relação ao período total de planejamento. Podemos afirmar que a solução é viavel e ótima até o período de planejamento equivalente a ocorrência do último conflito resolvido.

- Algoritmo com Sucessivas Buscas em Profundidade:

Neste caso, o processo decisório fica restrito aos seguintes passos principais:

1 - Seleção do estado com melhor preferência (situado no topo da lista) entre os estados abertos (contextos). Caso a lista esteja

vazia termine ;

2 - Seleção de operadores segundo o contexto escolhido (análise de conflitos);

3 - Determinação de preferências no conjunto de operadores aplicáveis em função do conhecimento intensivo (restrições , heurísticas, etc);

4 - Avaliação dos operadores (semântica de preferências) segundo os objetivos do problema e eliminação dos operadores segundo os critérios de poda e o limite inferior existente;

5 - Geração paralela de estados sucessivos segundo a aplicação dos operadores remanescentes. Em caso de falha volte ao passo 1;

6 - Inserção dos estados gerados no espaço do problema (lista de abertos) em função do valor de avaliação;

7 - Realização de uma busca em profundidade no estado escolhido, envolvendo:

7.1 - Seleção de operadores segundo o contexto escolhido (análise de conflitos) . Caso o estado represente o estado final, termine a busca em profundidade. Atualize o limite inferior do problema e retorne ao passo 1 ("backtracking" não cronológico);

7.2 - Determinação de preferências no conjunto de operadores aplicáveis em função do conhecimento intensivo (restrições , heurísticas, etc);

7.3 - Avaliação dos operadores (semântica de preferências) segundo heurísticas,incluindo a verificação dos critérios de poda quanto a satisfação dos objetivos e verificação do limite inferior,determinando um operador com melhor preferência;

7.4 - Geração do estado sucessor segundo a aplicação do operador com melhor preferência. Em caso de falha retorne ao passo 1 ;

7.5 - Retorne ao passo 7.1;

Como o algoritmo examina todos os estados da lista de abertos , o mesmo, também, termina com uma solução ótima associada ao limite inferior do problema. Entretanto, é possível interrompermos a busca no passo 7.1, quando atingimos um limite pré-estabelecido (formulação de inequações).

A realização da busca em profundidade é interrompida se detectarmos um estado rejeitado (passo 7.2) ou imcompatível com o limite inferior do problema ou com os níveis mínimos de aspiração dos objetivos (passo 7.3). Neste caso, é realizado o "bactracking" ao melhor estado da lista de abertos.

É importante destacarmos que através da utilização deste algoritmo, temos sempre em mão uma solução viável e satisfatória , abrangento todo o período de planejamento.

- Algoritmo IDA*:

Neste caso, o algoritmo realiza sucessivas buscas em profundidade com realização de "bactracking " com dependência cronológica , iniciando no nível inicial, e , aumentando, gradativamente, o nível de profundidade, até atingir o nível final.

Como em cada busca, associada a um nível de profundidade, o algoritmo examina todos os estados , é possível estabelecer, iterativamente, um novo limite inferior para o problema. A cada iteração, o limite inferior é igual ao valor mínimo de todos os estados que excedem o limite inferior (viáveis) da iteração anterior. Como a cada iteração o limite inferior apresenta valores crescentes e a função de avaliação e monotonicamente decrescente para um problema de máximo, então o algoritmo determina na última iteração a solução ótima do problema.

Embora, segundo R.E. Korf [1985], este algoritmo seja , a exemplo do algoritmo A*, ótimo em termos de complexidade em tempo e espaço e valor da solução , sobre a classe de todos os algoritmos de busca em árvore admissíveis, o mesmo não teve um bom desempenho em nosso sistema. Ou seja, a obtenção de limites inferiores a cada iteração não compensou o esforço excessivo e redundante necessário a realização das sucessivas buscas em profundidade até atingirmos a profundidade final.

Principalmente, considerando que, em nossa implementação, o "bactracking" cronológico foi feito através do retorno ao estado raiz, dado a insuficiência de espaço para guardarmos os estados parciais.

6.1.5 - Aplicações de Arquiteturas Inteligentes

- Aplicação da Arquitetura SOAR ao Sistema R1/XCON:

A arquitetura SOAR é uma arquitetura proposta para a solução de uma larga classe de problemas que se adaptam as hipóteses fundamentais de arquiteturas inteligentes. Estes problemas incluem desde problemas de jogos computáveis até tarefas que exigem o emprego de conhecimento intensivo e dependente do domínio como a realização das funções de diagnóstico e projeto de configurações.

Entre as tarefas de conhecimento intensivo, destacamos a utilização da arquitetura SOAR no sistema R1, para a configuração de computadores VAX e PDP-11 na Digital Equipment Corp., onde o sistema é denominado XCON. Para tanto, desenvolveu-se um sistema denominado R1-SOAR (P.S. Rosenbloom e outros [1985]) que integra o conhecimento desenvolvido para o sistema R1 (J. McDermott [1982]) com a arquitetura geral de solução de problemas SOAR.

O sistema R1 emprega , aproximadamente, 3300 regras de produção particionadas em 321 sub-tarefas e possui um banco de dados contendo descrição de aproximadamente 7000 componentes. A respeito de uma descrição mais atualizada deste sistema, recomendamos consultar W.B. Rauch-Hindin[1985].

Por outro lado, R1-SOAR consiste de uma implementação de SOAR, que exibe 25% da funcionalidade do sistema R1, realizando a tarefa de configuração da "unibus", utilizando, para tanto, aproximadamente 1/6 do conhecimento de R1 , extraído das regras de produção transcritas na linguagem OPS5.

Para realizar esta tarefa , o sistema R1 executa aproximadamente 1000 regras , enquanto que, o sistema R1-SOAR, exige a execução de somente 90 regras para completar o mesmo serviço, tornando viável a utilização de uma arquitetura geral na solução de tarefas que exigem o emprego de conhecimento intensivo. Uma explicação mais detalhada do desenvolvimento e funcionamento do sistema R1-SOAR , bem como , de sua utilização, está mostrada no trabalho de P.S. Rosenbloom e outros [1985].

- Aplicação da Arquitetura SIGT as Funções de Planejamento e "Scheduling":

A arquitetura SIGT é uma arquitetura específica, voltada para a realização de planejamento, tendo como requisito básico, capacidade para suportar o gerenciamento inteligente de trens em tempo real.

As tarefas associadas ao SIGT, são bem mais complexas que as tarefas de solução de jogos e projeto de configurações , no sentido de envolverem a realização de uma busca heurística admissível que satisfaça um conjunto de objetivos e um conjunto de restrições estruturais de um problema de "scheduling", utilizarem conhecimento intensivo, empregarem raciocínio temporal e possuírem restrições associadas a capacidade de operação em tempo real.

A arquitetura SIGT é composta de uma base de conhecimento e de um sistema de produção que foram apresentados em capítulos anteriores, e constituem uma proposta para o desenvolvimento de sistemas de conhecimento intensivo com aplicações em tempo real. Entretanto, a operacionalização do espaço do problema , incluindo :

- algorítmos heurísticos de busca;
- estruturas de representação de estados, incluindo tabela de tempos;
- funções de identificação de conflitos;
- funções de transformação ou operadores;
- funções para o cálculo de intervalos e atualização do conhecimento(raciocínio temporal);
- funções para a geração de estados (atualização da tabela de tempos);
- desenvolvimento de objetivos, e
- funções para a avaliação de operadores,

tiveram uma implementação específica, segundo o desenvolvimento de tipos abstratos de dados, pois seria impraticável , dado ao grau de exigência e complexidade , a representação destas funções, procedimentos e estruturas em um sistema de produção , a exemplo da arquitetura SOAR.

Desta forma, quanto a realização do gerenciamento inteligente do tráfego de trens, podemos destacar algumas vantagens da arquitetura SIGT

quando aplicada a execução de planejamento, em relação a arquitetura SOAR:

1 - O conhecimento intensivo é representado diretamente em produções;

2 - O controle do conhecimento ou as funções de controle de busca são independentes do domínio e associados, principalmente, ao método de busca empregado e a estrutura do problema;

3 - Representação do conhecimento declarativo e relacional, de longo prazo, em "frames";

4 - Capacidade para raciocínio temporal;

5 - Capacidade para processamento aproximado;

6 - Incorpora modelos de programação matemática "fuzzy" que permite a geração de multi-objetivos associados a funções de pertinência que avaliam o nível de satisfação das decisões, e determinam uma semantica de preferência em operadores segundo os princípios básicos da racionalidade;

7 - Incorpora modelos de "scheduling" que orientam a formação do espaço do problema e a transformação de estados, permitindo a realização de uma busca orientada segundo as restrições estruturais destes modelos;

8 - Incorpora um modelo de simulação que permite o desenvolvimento de novas produções, atraves do exercício contínuo de operações simuladas;

9 - Incorpora um modelo de monitoração que permite avaliar a qualidade dos planos e a viabilidade do gerenciamento em tempo real em função, principalmente, do nível de imprecisão do sistema.

6.1.6 - Aprendizado

- Teoria "Chunking":

O mecanismo de aprendizado baseado na teoria "chunking" (P.S. Rosenbloom e A. Newell [1986]) associado a arquitetura SOAR, consiste em acrescentarmos a memória de longo prazo ou a base de regras , novas produções que resumem o processamento de um sub-objetivo associado a um espaço de problema.

Como a geração de um sub-objetivo está, automaticamente, associada a ocorrência de ímpasses e a ocorrência de ímpasses é

ocasionada pela falta de conhecimento intensivo, podemos afirmar que o processo de "chunking" e um processo permanente e contínuo de aprendizado no sentido de incorporar o conhecimento necessário a solução de situações de ímpasses, através de um processo de generalização. Desta forma, sempre que o sistema depara , futuramente, com situações similares aquelas que levaram a criação das produções , a solução dos ímpasses e realizada diretamente, através da aplicação do conhecimento adquirido.

A teoria "chunking" se baseia no fato de que a performance na execução de tarefas melhora com a prática, ou seja, o tempo necessário a execução da tarefa decresce a medida em que executamos a mesma maior número de vezes, ou a medida em que adquirimos mais experiência.

O conhecimento incorporado ao longo do tempo , fruto da experiência na execução sucessiva de tarefas, é considerado como um conhecimento especialista .

Portanto, se um sistema tem capacidade de gerar,automaticamente, novas produções e , futuramente , reconhecer situações onde estas produções se aplicam , podemos dizer que este sistema se torna cada vez mais inteligente no sentido de incorporar gradativamente mais conhecimento especialista.

Se observarmos, na arquitetura SOAR, a solução de ímpasses através da geração automática de sub-objetivos, podemos afirmar que o mecanismo "chunking" tem uma conduta "bottom-up", pois novas produções são incorporadas a base de regras a partir da solução de objetivos terminais , ou seja, objetivos que não determinam a geração de sub-objetivos (lembrando que a solução de ímpasses pode estabelecer uma hierarquia de sub-objetivos).

Portanto, a capacidade de aprendizado da arquitetura SOAR está , íntimamente, associada a capacidade da arquitetura em reconhecer situações de ímpasse e determinar a geração automática de sub-objetivos.

Como exemplo, podemos afirmar que a solução de um ímpasse, relativo a determinação de preferência entre operadores, podera levar a criação de uma produção que, posteriormente, irá controlar com mais eficiência o processo de busca. Caso, na solução de uma tarefa, a aplicação de um operador resulte em uma situação de "não troca" , ou seja, não ha modificações na estrutura de um contexto, o mecanismo

"chunking" determinará a criação de uma produção que estabelecerá a rejeição do mesmo operador em situações similares.

- Simulação de Produções:

Em nosso sistema, apesar de não contarmos com um mecanismo automático de aprendizado, a ampliação do conhecimento do sistema é possível, através do desenvolvimento de produções para a solução de situações de ímpasse. As maiores restrições a geração automática de produções são:

1 - Na arquitetura SIGT, o espaço do problema não é implementado na forma de um sistema de produção. Desta forma, não é possível ao sistema empregar uma conduta "bottom-up" no processo de aprendizado, no sentido de recuperar os elementos da memória de trabalho, necessários a realização dos sub-objetivos, até o objetivo terminal;

2 - Na arquitetura SIGT, a solução de sub-problemas que retratam a existência de ímpasses, é feita utilizando-se o conhecimento representado em um sistema de produção. Portanto, se o conhecimento não for suficiente, o ímpasse não será resolvido.

Desta forma, podemos afirmar que a arquitetura não possui capacidade para a aquisição interna de conhecimento, mas somente, capacidade para avaliar se o conhecimento existente é necessário a solução de situações de ímpasse.

A primeira vista, esta deficiência parece ser muito prejudicial ao bom desempenho de uma arquitetura inteligente voltada para a solução de uma classe geral de problemas. Entretanto, este não é o caso da arquitetura SIGT. Na realização do controle e gerenciamento dos trens, a aplicação do conhecimento prático adquirido, fruto da experiência de trabalho na elaboração de planos, é menos expressiva que o desenvolvimento de linhas de raciocínio associadas as condições de viabilidade e otimalidade do planejamento, como vimos na seção 2.3.4.

O sistema SIGT, se desenvolvido segundo a arquitetura SOAR, certamente não teria capacidade e flexibilidade suficiente, para a elaboração de "schedules" ótimos em um tempo de resposta restrito.

A geração de novas produções para o sistema, como foi apresentado no capítulo V, é fruto de uma análise minuciosa das situações de conflito que, potencialmente, levam a situações de ímpasses. Desta forma, devemos observar, atentamente, as condições de viabilidade e otimalidade do problema segundo a árvore decisória resultante da aplicação sucessiva de operadores, e estabelecermos heurística apropriadas que, adequadamente, determinem preferências na solução de ímpasses.

Para tanto, foi de fundamental importância o desenvolvimento de um simulador para o SIGT, o qual possibilitou a avaliação parcial de operadores , simulação de novas produções e simulação de linhas de raciocínio na solução de ímpasses.

Outra importante fonte de informação a respeito da eficiência do processo decisório, que pode ser útil ao desenvolvimento de novas produções, consiste em analizarmos nos estados finais , todas as decisões e respectivas situações de conflito que se efetuaram ao longo do processo de planejamento.

6.2 - Técnicas de Decomposição

Segundo R.E. Korf [1987], o problema de planejamento é visto como a solução de um problema de busca . Para tanto, afim de reduzir a complexidade em tempo e espaço deste processo, Korf sugere a utilização de mecanismos de controle do conhecimento baseados na decomposição de sub-objetivos, decomposição e abstração de espaços e uso de macro-operadores.

Um algorítmo heurístico que utiliza uma função de avaliação heurística, durante o processo de busca, diminui, consideravelmente, a complexidade em tempo e espaço no sentido de permitir uma redução do fator de ramificação. Entretanto, a função de complexidade em tempo do algorítmo continua sendo de ordem máxima exponencial. Portanto, afim de obtermos resultados mais eficientes, sem o sacrifício da otimalidade, é necessário desenvolvermos outras formas de controle do conhecimento.

O uso apropriado de sub-objetivos, a exemplo da arquitetura

SOAR, reduz, consideravelmente , o esforço de busca empregado na tarefa de solução de problemas . A principal razão é que, decompondo um problema exponencial, em dois sub-problemas , reduzimos, quase pela metade, a complexidade da busca.

6.2.1 - Decomposição Hierárquica:

No sistema de produção desenvolvido para o SIGT, adotamos um esquema de decomposição de sub-objetivos na forma hierárquica, em três níveis, estabelecendo, desta forma, um conjunto de sub-objetivos independentes (quando situados no memo nível da hierarquia ou quando não possuirem dependência) e outro conjunto de sub-objetivos sequenciáveis e não independentes (para o caso contrário).

Considerando a complexidade de cada módulo de regras como uma função linear do número total de regras e meta-regras existentes no módulo, teremos uma complexidade total em função do número de regras executadas, considerando o pior caso, da ordem de :

$$N_3 \times N_2 \times N_1 + N_4 \times N_1 + N_5 \ , \qquad (120)$$

onde:

N_3 é o número de regras do maior módulo de regras situado no nível 3 da hierarquia;

N_2 é o número de meta-regras do maior módulo de regras situado no nível 2 da hierarquia;

N_1 é o número de meta-regras do maior módulo de regras situado no nível 1 da hierarquia,

N_4 é o número de regras situadas no módulo escolhido do nível 2 da hierarquia, e

N_5 é o número de regras situadas no módulo escolhido do nível 1 da hierarquia.

Entretanto, se considerarmos que no primeiro nível , determinamos a execução de uma única meta-regra relativa ao tipo de conflito existente , e no nível medio , a execução de duas meta-regras relativa as condições de satisfabilidade de dois operadores, e , também, que no nível 1 da hierarquia temos a existência de somente um módulo, e

que somente um módulo é escolhido em cada nível , durante o processo decisório, teremos a nova complexidade da ordem máxima de :

$$Max \{N_3 \times 2 + N_4 + N_5\} . \qquad (121)$$

Lógicamente, a expresssão de complexidade se refere aquela que fornecerá o maior valor possível quanto as possíveis dependências entre módulos, situados nos diferentes níveis da hierarquia.

6.2.2 - Decomposição em Sub-objetivos não Sequenciáveis

Segundo R.E. Korf[1987], sub-objetivos não sequenciáveis formam um conjunto de sub-objetivos não independentes que são afetados, mútuamente, no sentido de propiciarem um progresso em relação ao objetivo principal.

No SIGT este objetivo vem a ser, exatamente, a conjunção ou interseção lógica dos sub-objetivos associados a um conjunto de funções de pertinência que avaliam os respectivos níveis de satisfação, na formulação multi-objetiva "fuzzy" do problema. No capítulo III , verificamos as contribuições desta forma de decomposição ao processo de solução do problema.

Outra forma de decomposição existente no SIGT, está relacionada ao estabelecimento de um conjunto de sub-objetivos não independentes e sequenciáveis, associados as dimensões do planejamento. Neste sentido, a a exemplo da decomposição definida para o sistema de produção , é possível resolvermos os sub-objetivos sequencialmente, desde que seja estabelecido um critério lógico de ordenamento.

Primeiramente, podemos observar este tipo de decomposição no processo de desenvolvimento de planos consecutivos . Neste caso, dividimos o período de planejamento em intervalos , determinando um conjunto de sub-objetivos. Este tipo de decomposição afeta a otimalidade temporal da solução, ou seja, segundo R.E. Korf, o caminho da solução ótima associado ao objetivo único é melhor do que a soma dos custos associados a cada sub-objetivo. Entretanto, para efeito da redução da complexidade diminuimos, enormemente, o número de conflitos potenciais para dado comprimento linear de planejamento e,

consequentemente, o nível de profundidade do processo de busca.

Esta forma de decomposição é totalmente necessária e está associada a necessidade de replanejamento em função do grau de imprecisão do sistema.

Em segundo lugar, podemos destacar o desenvolvimento de planos adjacentes, correspondentes a cada setor do sistema ferroviário, que afeta a otimalidade linear do planejamento. Da mesma forma, esta forma de decomposição diminui o número de conflitos potenciais para um dado intervalo de planejamento e, consequentemente, o nível de profundidade do processo de busca. Temos, também, uma redução proporcional das operações realizadas sobre a tabela de tempos , bem como do espaço de memória requerido, já que a mesma possui como uma dimensão, o número de seções sob planejamento.

Esta forma de decomposição é totalmente necessária em função da restrição de memória existente para a lista de estados abertos e, também, devido a divisão lógica da ferrovia em setores controláveis. Entretanto, a integração de planos adjacentes não é imediata , como no caso anterior, requerendo o desenvolvimento de um sistema de coordenação, que permita a integração dos planos existentes.

6.2.3 - Macro-Operadores

Segundo R.E. Korf[1985], macro-operadores consistem de uma sequência de operadores primitivos, utilizados no sentido de resolver mais de uma instância do problema. Desta forma, podemos associar o desenvolvimento de macro-operadores a uma forma de aprendizado , baseada, também, na idéia de generalização de soluções particulares de problemas associadas a sub-objetivos específicos.

Ou seja, durante a solução de um problema computável, como o jogo dos oito ou o jogo da velha, nos preocupamos, essencialmente , em aprender ou desenvolver uma estratégia de solução do problema guardando uma sequência de operadores. Desta forma, futuramente, através da aplicação desta sequência de operadores, é possivel obtermos a solução de outras instâncias do problema ou jogarmos outras partidas com maior facilidade, realizando, consequentemente, menor esforço de busca.

É interessante observarmos, que a utilização de macro-operadores em certos tipos de problemas com a propriedade de decomposição dos operadores, permite atingirmos a solução, sem a utilização de busca. Como exemplo, podemos citar o jogo dos oito, para o qual R.E. Korf desenvolve um conjunto de 35 macro-operadores organizados em uma tabela , suficientes para a solução de qualquer instância do problema.

A técnica de solução consiste em aplicarmos sucessivamente (o próximo sub-objetivo determina a escolha da próxima macro) cada macro-operador, no sentido de satisfazermos um sub-objetivo que corresponde a colocação sucessiva de cada peça do jogo em seu devido lugar. Caso a colocação de uma nova peça provoque , temporáriamente, uma alteração das peças já colocadas anteriormente, o macro-operador associado restabelece os posicionamentos corretos. Ou seja, apesar da existência de um conjunto de sub-objetivos não sequenciáveis em relação aos operadores primitivos, é possível estabelecer um conjunto de macro-operadores que resgata esta propriedade. Portanto, a questão de sequenciabilidade do conjunto de sub-objetivos passa a ser dependente do conjunto de operadores aplicados ao problema.

Como um macro-operador corresponde a um conjunto de ações sequênciais , a sua utilização no sistema SIGT é possível, mas viável, somente, na solução de sub-problemas específicos referentes a solução de ímpasses, por razões já, amplamente discutidas,ver seções 2.2.4 e 6.1.6.

6.3 - Integração entre Técnicas de Inteligência Artificial e Pesquisa Operacional

6.3.1 - Resultados Comparativos

Comparado aos modelos matemáticos utilizados para a geração de planos de horários, o SIGT apresenta uma performance extremamente superior, possibilitando, desta forma, a realização do planejamento dinâmico necessário ao controle e gerenciamento de trens em tempo real.

Nas figuras 50 e 51, temos a demonstração de duas grades de trens relativas ao planejamento tático efetuado pelo SIGT para o controle e gerenciamento de 6 trens em um trecho ferroviário,

compreendendo 10 seções singelas. Na figura 50, a realização do planejamento foi feita sem a utilização do sistema de produção. A solução encontrada, com o uso do algoritmo A* e formulação Max-Min, é a solução ótima do problema de programação multiobjetiva "fuzzy" , determinando um valor máximo para a função de avaliação igual a 0.8302.

A figura 51, retrata a realização do mesmo planejamento, porém, com a utilização do sistema de produção. Podemos observar que a solução encontrada, com valor igual a 0.8319, foi superior a primeira, com tempo de computação quase 3 vezes inferior. O sistema de produções sugeriu, durante o processo de solução do problema, a formação de uma onda de trens (trens 2 e 3) com prioridade de avanço nos cruzamentos.

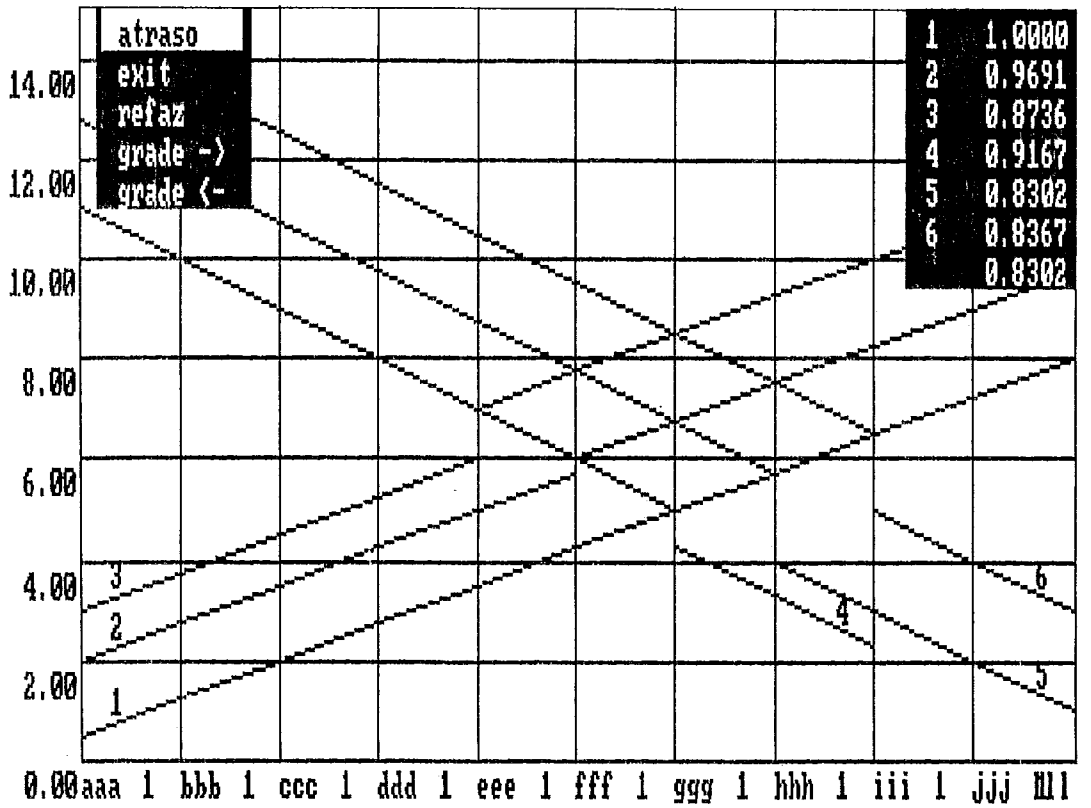


figura 50: plano elaborado sem o sistema de produção.

6.3.2 - Justificativas

Este resultado, surpreendente, comprova a vantagem do uso de técnicas de Inteligencia Artificial em relação as técnicas tradicionais de Pesquisa Operacional. Segundo observações feitas po Mark S. Fox [1989], podemos destacar seis razões principais que justificam o uso de

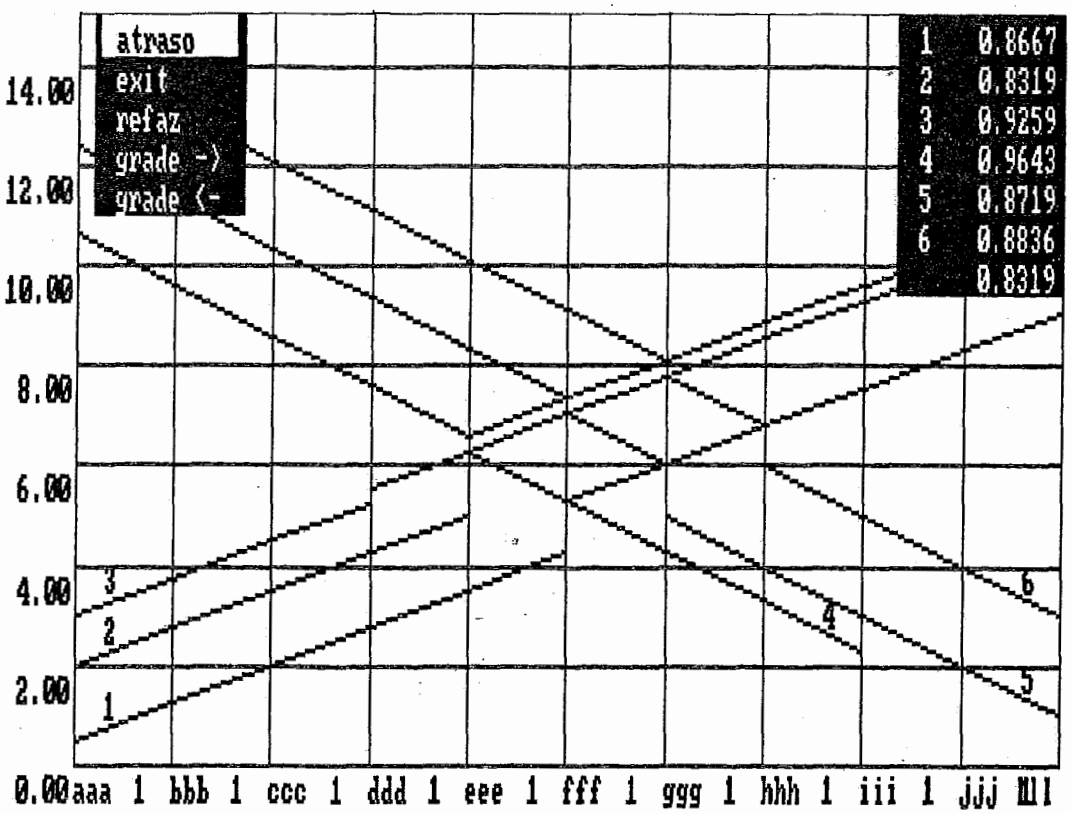


figura 51: plano elaborado com o sistema de produção.

técnicas de Inteligência Artificial no processo de modelagem e solução de problemas:

1 - O processo de representação em Inteligência Artificial extende o processo de representação em Pesquisa Operacional pelo uso de abstrações e diferenciação, no sentido de ser uma abstração qualitativa de um modelo quantitativo. Como exemplo, podemos citar a utilização de um conjunto de funções para cálculo de intervalos ou um modelo de tempo relacional que permite a verificação das restrições de capacidade durante o problema de planejamento e "scheduling";

2 - O processo de busca é um método natural empregado tanto em métodos de solução de problemas em Inteligência Artificial quanto em Pesquisa Operacional;

3 - A Inteligência Artificial extende o método de solução de problemas através da aplicação oportunista do conhecimento. Na Pesquisa Operacional a aplicação do conhecimento é mecanicista. Como exemplo, podemos afirmar que a incorporação do conhecimento na forma de restrições em um modelo matemático aumenta o tamanho da instância do

problema, e, consequentemente, a complexidade em tempo e espaço do algoritmo utilizado para resolvê-lo, justificando , na Programação Matemática, a maior dificuldade na solução dos modelos de programação inteira comparados a programação linear (neste caso, a técnica mais comum de solução , consiste em acrescentarmos a formulação relaxada do problema planos de corte ou restrição que garantam a integralidade das variáveis).

Por outro lado, a incorporação de conhecimento oportunista a base de conhecimento, propicia, em nossa arquitetura, a determinação de preferências em operadores, reduzindo o fator de ramicação e , consequentemente, o processo de busca utilizado para a solução do problema. Como exemplo, citamos um simples algoritmo que progride através de comparações sucessivas, ou seleção e teste. Neste caso, quanto maior for o número de informações a respeito do estado final, mais seletivo e rápido se torna o processo de busca;

4 - A Inteligência Artificial extende, segundo H. Simon [1987], o método de solução de problemas da análise para o projeto. No nosso exemplo, ultrapassando as técnicas tradicionais de análise, o sistema projetou a formação de uma onda de trens , retardando préviamente o trem 9 na seção ddd-eee. Esta decisão, que alterou o equacionamento matemático do problema na forma "job-shop scheduling", permitiu a obtenção de uma solução superior;

5 - A Inteligência Artificial extende o método de solução de problemas, incluindo a formulação e re-formulação automática do mesmo. Quando utilizamos um processo de busca orientado pelas restrições do problema, podemos, através de pré e pós-análises no espaço de estados, alterar a formulação pela modificação de restrições e seleção de operadores. No nosso exemplo, a tomada de decisão que determinou a formação da onda de trens só foi possível devido a capacidade de pós-analise do algoritmo A*, e finalmente,

6 - A Inteligência Artificial extende o método de solução de problemas através do aprendizado ou da inclusão, ao longo das experiências, de conhecimento especialista . Neste sentido, podemos destacar o uso do mecanismo "chunking" em SOAR ou o desenvolvimento de macro-operadores como sugerido por R.E. Korf[1985].

Não achamos, entretanto, que a aparente superioridade e maior

flexibilidade das técnicas de Inteligencia Artificial sobre os métodos tradicionais de análise, vão levar a um abandono das técnicas de Pesquisa Operacional.

Ao contrário, conforme H. Simon [1987] definiu o título de seu trabalho: " duas cabeças pensam melhor do que uma ", e como estamos verificando, a colaboração entre a Inteligência Artificial e a Pesquisa Operacional na solução de problemas, é totalmente necessária, oportuna e produtiva.

6.4 - Técnicas de Simulação

Nesta seção, apresentamos os principais estilos de simulação quanto a forma de programação e controle de eventos, bem como, o desenvolvimento de uma base teórica para o modelo de simulação incorporado ao nosso sistema de controle e gerenciamento de trens.

Tratamos, basicamente, de sistemas de simulação por eventos discretos . Entretanto, em nosso modelo , permitimos um alto grau de paralelismo na realização de eventos, e, também, incorporamos grande flexibilidade nos mecanismos de programação e controle que regulam o avanço da simulação, através do uso de técnicas de Inteligência Artificial.

Na simulação por eventos discretos, o modelo de simulação admite a troca de estados somente em períodos discretos de tempo, ao contrário da simulação contínua na qual a troca de estados é efetuada continuamente. A representação mais geral de um modelo de simulação é baseada na modelagem entidade-evento, a qual representa o sistema como um conjunto de entidades, permitindo a alteração dos atributos das entidades em função da realização de eventos associados a ocorrência de atividades.

A troca de estados em um sistema simulado, corresponde a execução de eventos que são organizados sequencialmente em uma lista, denominada lista de eventos, segundo a ordem cronológica.

6.4.1 - Simulação Orientada pela Programação de Eventos

- Funcionamento:

Este tipo de simulação é direcionado segundo a realização e programação de eventos . A ordenação ou sequenciamento cronológico de eventos, implica na existência de um relógio que controla o tempo do sistema a ser simulado. Neste caso, o relógio é avançado segundo intervalos discretos de tempo, equivalentes ao período de ocorrência dos eventos. Desta forma, cada evento realizado é , em seguida, programado e inserido na lista de eventos na ordem apropriada.

A lista de eventos, de um modo geral, tem a forma :

$$(e_1,t_1),\ (e_2,t_2),\ldots,(e_n,t_n)\ , \qquad (122)$$

onde: e_i representa o i-ésimo evento e t_i o tempo de ocorrência associado , observando que:

$$t_1 \leq t_2 \leq \ldots \leq t_n \qquad (123)$$

A um nível abstrato, D. Kumar [1986] associa a cada evento programado, uma upla formada por:

$$(\ e_i,\ t_i,\ j,\ s\ ), \qquad (124)$$

onde:

"j" representa a entidade que provoca o evento e

"s" representa o conjuto de entidades afetadas pela realização do evento.

O processamento ou a realização de um evento e_i pode ou não determinar a programação de outro evento em um instante "t", bem como o cancelamento ou a suspensão temporária de outros eventos existentes na lista. Como a lista de eventos apresenta um ordenamento crescente de valores associados ao tempo do sistema, a simulação terminará, sempre, com o processamento de um número finito de eventos, dentro de um período fixo de tempo, conhecido como período de simulação.

- Algorítmo de Controle e Programação:

O algorítmo geral deste estilo de simulação tem a forma :

```
Início
  termina <- FALSE;
  TempoSimulação <- 0;
  Enquanto não (termina ) Faça
    Escolha o evento de tempo mínimo da lista de eventos,
    ou seja , obtenha uma upla ( e_i, t_i , j, s ) ;
    Avançe o relógio da simulação , TempoSimulação <- t_i ;
    Se TempoSimulação > TempoTotal Então
      termina <- TRUE
    Senão
      Para cada entidade k pertencente a s Faça
```

```
            Modifique atributos se necessário;
            Programe, se necessário, o(s) evento(s) causados por k ;
          FimPara;
          Programe, se necessário,  o(s) evento(s) causados por j;
          Insira os eventos na lista de eventos segundo a ordem
          cronológica;
        FimSe;
      FimEnquanto;
    Fim.
```

- Aspectos Negativos:

Os principais aspectos negativos deste estilo de simulação estão relacionados a existência da lista de eventos , que pode conter um número de ocorrências muito elevado, tornando ineficientes as operações de inserção, que são, em geral, de complexidade linear para listas encadeadas (as implementações mais eficientes como a implementação de listas com estruturas de dois níveis (WR. Franta e K. Maly [1977]), baseada no algorítmo de listas indexadas (J.G. Vaucher e P. Durval [1975]) e na noção de árvores balanceadas (W.R. Franta e K. Maly [1977]) apresentam complexidade no pior caso da ordem $O(\sqrt{n})$, onde "n" se refere ao número de eventos na lista).

Portanto, a existência de uma técnica melhor de simulação , implicaria na redução do número de inserções de eventos, ou, também, na possível simulação paralela de eventos. Entretanto , dado ao fato de que , para garantirmos o processamento de um evento, temos que esperar que todos os eventos programados anteriormente terminem, e que o único evento garantido é aquele com tempo de realização mínimo na lista de eventos, fica difícil estabelecermos algum tipo de melhora.

A justificativa a este raciocínio, se deve ao fato de que um evento simulado mais tarde com término no instante t_1, pode afetar outro evento simulado mais cedo com término no instante t_2, $t_2 > t_1$, tornando o mesmo incorreto.

Da mesma forma, a programação de um evento para o instante t_1, correspondente ao tempo mínimo da lista de eventos, pode ficar afetada, se, em seguida, um outro evento determinar a programação de um próximo evento para um instante t_2, $t_2 < t_1$.

6.4.2 - Simulação Orientada por Processos

- Funcionamento:

No estilo de simulação anterior, a realização de eventos causa sucessivas alterações as entidades associadas, através da modificação de atributos, constituindo, desta forma, o mecanismo de controle responsável pelo avanço da simulação. Na simulação orientada por processos, o mecanismo de controle, bem como , a modelagem do sistema, são um pouco diferentes.

Neste estilo de simulação, a modelagem do sistema é realizada através da criação de processos. Segundo definição de M.H. MacDougall e J.S. MacAlpim [1973] , processos são:

" Entidades dinâmicas ou instâncias pertencentes a um conjunto de atividades relacionadas lógicamente. Portanto, um processo é definido por uma classe associada a um conjunto de pares atributos-valores particulares (instância) e sua conduta descrita por um conjunto de regras (que descrevem as atividades de todos os processos de uma mesma classe)."

Em um ambiente de concorrência, os processos interagem entre si, disputando recursos e trabalham, cooperativamente, exigindo um conjunto de mecanismos de sincronização, como forma de garantir a integridade e sequenciamento do sistema, bem como as propriedades conhecidas como "safety" e "liveness" .

Para tanto, devemos ter:

1 - Em relação a garantia de integridade e sequenciamento do sistema, um mecanismo que estabeleça um ordenamento de operações segundo a restrição cronológica (sequenciamento de eventos no tempo), observando o relógio do sistema. Esta restrição esta associada a existência de interação entre processos.

2 - Em relação a propriedade de "safety" , primitivas de sincronização que permitam uma exclusão mútua ou exclusivo acesso (B. Hansen [1972]) em relação aos recursos do sistema , observando as regiões críticas de cada processo e, finalmente,

3 - Em relação a propriedade de "liveness" , garantias de que o processo de simulação termine. Ou seja, não podemos ter situações de "loop" infinito ou "deadlock" (B. Hansen [1973]);

Como exemplo desta nova filosofia, vamos definir um modelo de simulação , para a operação de um pequeno trecho de uma ferrovia. Na visão orientada por eventos , teríamos como componentes do sistema:

entidades : seções da ferrovia;
filas : patios de cruzamento, e
agentes associados aos eventos : trens .

Teríamos, tambem, como principais eventos:

- saída de um trem de um pátio$_j$;
- chegada de um trem ao próximo pátio$_{j+1}$, caracterizando a atividade de percurso associada ao trem e a entidade seção entre os pátios citados;
- ocupação do pátio$_j$ de cruzamento pelo trem ;
- desocupação do pátio$_j$ de cruzamento pelo trem, caracterizando a atividade de espera associada ao trem e ao pátio$_j$ de cruzamento, e
- solução do conflito de precedência entre dois trens em uma seção da ferrovia em função da disputa de recursos e do tipo de atendimento realizado.

Desta forma, podemos considerar a simulação de um sistema ferroviário como a simulação sequencial de um conjunto de entidades interligadas que representam atividades de serviços (seções) associadas a filas de espera (pátios) em função da chegada de trens.

Portanto , o processo de simulação irá depender fundamentalmente:

- da interligação do sistema;
- do tipo de serviço ou atendimento associado a cada entidade;
- da distribuição do tempo de serviço;
- da capacidade das filas de espera e
- do processo associado a geração de novos trens.

Na visão de simulação orientada por processos, teremos na modelagem do sistema dois processos básicos :

. um processo de atendimento, e

. um processo de chegada .

Ou seja, cada entidade (seção) do sistema a ser modelado fica representada por dois processos básicos , um processo de atendimento e um processo de chegada, que caracterizam as atividades de percurso de um trem ao longo de uma seção. É evidente , que a cada processo devemos associar uma fila de espera com limites de capacidade de forma a representar a abstração dos pátios de cruzamento.

Portanto, a simulação do sistema fica caracterizada pela interação entre os vários processos, devendo, principalmente, observar a ocorrência de conflitos provocada pela disputa momentânea de uma seção da ferrovia por dois trens. Neste caso, ao darmos prioridade de atendimento ao primeiro trem, o processo de chegada deve considerar a existência de uma seção crítica no processo de atendimento, colocando o segundo trem na fila de espera.

Temos que observar, também, a garantia da integridade e sequenciamento do sistema, ou seja, a interação entre processos deve observar o relógio do sistema. Lógicamente, se os processos fossem totalmente assíncronos, naõ haveria interação e, consequentemente, não teríamos a necessidade de relógio.

- Implementação com SIMULA-67:

Para possibilitar a realização deste estilo de simulação, surgiu a primeira linguagem de simulação orientada por processos SIMULA (O.-J. Dahl e K. Nygaard [1966]), depois SIMULA 67 (G. M. Birtwhistle e outros [1973]), baseadas em ALGOL, que incorporou um conjunto de construções que permitiram :

. fazer especificações e declarações de classes;

. gerar instâncias de classes ou processos (objetos);

. concatenar declarações de classes permitindo o desenvolvimento estruturas hierárquicas e

. possibilitar a transferência de controle entre processos e o programa principal, estabelecendo o uso de co-rotinas.

A linguagem SIMULA incorporou, também, um conjunto de

construções visando facilitar o desenvolvimento de sistemas de simulação, incluindo:

- uma classe voltada para o processamento de listas;
- uma classe de simulação que é uma sub-classe da classe lista , voltada para a simulação do sistema;
- mecanismos para a geração, manipulação e cancelamento de eventos;
- relógio do sistema;
- lista de eventos sequenciais;
- procedimentos para a alteração do "status" de processos (objetos), e
- procedimentos para a sincronização de processos (objetos) com o relógio do sistema e a lista de eventos sequenciais.

Para um estudo mais detalhado da simulação por processos, utilizando a linguagem SIMULA, sugerímos o trabalho de W.R. Franta [1977]. Para nós, entretanto, é importante discutirmos como a linguagem SIMULA garante a integridade e sequenciamento da simulação através do uso de suas construções.

Dada a existência da iteração entre processos, e a necessidade de um relógio do sistema, a simulação por processos requer também o uso de uma lista de eventos sequenciais como forma de garantir a execução das diversas operações associadas a cada processo, respeitando a ordem cronológica. Esta lista de eventos, na linguagem SIMULA, tem a forma de uma lista duplamente encadeada onde cada elemento possui:

- dois ponteiros succ e pred;
- tempo de ocorrência do evento e
- ponteiro para o processo associado.

Inicialmente, o sistema a ser simulado e modelado como um conjunto de processos. A cada processo está associada uma determinada atividade, como exemplo : geração de chegadas ou serviços de atendimento para a simulação de uma fila. A realização das atividades , observando-se a interação entre processos , é conseguida através da programação de eventos. Como exemplo :

1 - A atividade de atendimento é caracterizada pelos eventos que determinam o início e o fim do atendimento. Desta forma , assim que um

processo de atendimento é acordado, e portanto inicia a atividade , automaticamente, o mesmo é suspenso até que a atividade ou a realização do serviço termine. Isto é conseguido através da programação de um evento na lista com tempo de ocorrência igual ao tempo equivalente ao término da atividade. Portanto, assim que o evento é processado, é realizado, automaticamente, a liberação do processo, ou seja, o processo está livre para iniciar novo atendimento ou , caso a fila associada esteja vazia, ser acordado pelo processo que provoca a geração de chegadas.

2 - A atividade de geração de chegadas fica caracterizada pelo intervalo de tempo que separa duas chegadas consecutivas. Desta forma , o processo de geração de chegadas programa um evento com tempo de ocorrência igual ao tempo equivalente ao tempo da próxima chegada. Portanto, assim que o evento é processado, é verificada a fila associada ao processo de atendimento; se a fila estiver vazia, o processo de geração de chegadas acorda o processo de atendimento; caso contrário, coloca-se mais um na fila de espera.

Portanto, através da programação de eventos associados a ocorrência de atividades, é possível manter a integridade e o sequenciamento do sistema.

Segundo Franta, a linguagem SIMULA admite as seguintes transições e possíveis estados para os processos (objetos), segundo a programação de eventos (ativar, reter, cancelar, etc.), conforme mostra a figura 52.

Estes estados, são caracterizados, respectivamente, por:

Ativo: o processo está em execução;

Suspenso: o processo está aguardando, e existe um evento associado na lista de eventos para a sua ativação;

Terminado: o processo não altera mais o seu estado até o fim da simulação, e

Passivo: o processo está aguardando, mas a sua reativação depende de outro processo. Portanto, não existe uma ativação programada como no estado Suspenso.

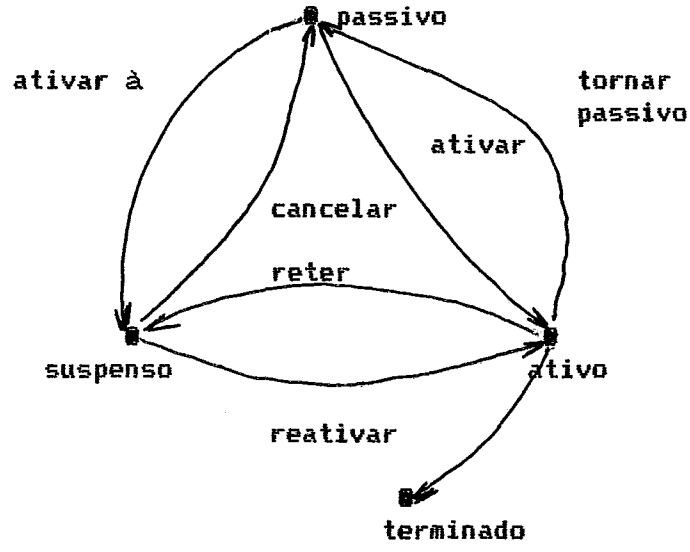


figura 52: transições e possíveis estados.

Sendo assím, podemos concluir que o processamento de um evento do tipo ativar, reter, cancelar, reativar, etc., está relacionado a sincronização de processos interativos com o relógio do sistema , e provoca a transição entre estados nos processos associados. Entretanto, como cada evento está associado a um único processo e cada evento é processado por vez; e considerando, também, que o mecanismo básico de sincronização da linguagem SIMULA é baseado na transferência de controle entre co-rotinas, podemos afirmar que não conseguimos, ainda, a simulação paralela de eventos.

Porém, é inquestionável, que este novo estilo de simulação representa um avanço considerável em relação a modelagem de sistemas dinâmicos, possibilitando o desenvolvimento do conceito de classes, instâncias, hierarquia de classes e transferência de controle entre processos. Estes conceitos foram considerados fundamentais no desenvolvimento de novos paradígmas de programação.

- Implementação com ADA:

R.M. Bryant [1982] apresenta um sistema de simulação orientado por processos implementado na linguagem ADA. O sistema utiliza as facilidades existentes na linguagem em relação a definição de processos por meio de tarefas ("tasks"), a criação de tipos abstrato de dados ou módulos de encapsulamento por meio de "packages", bem como a utilização

de primitivas existentes (mecanismo de "rendezvous", seleção não determinística, etc. (G.W. Cherry [1984])) como forma de possibilitar a sincronização entre processos.

O mecanismo básico de controle da simulação proposto na linguagem SIMULA, bem como, no sistema de R.M. Bryant [1982] é baseado na implementação de um procedimento responsável pela retenção do processo (programação de evento), ou seja, de um procedimento responsável pela transição de estados ativo -> passivo , bem como , de um processo responsável pela reativação de processos, ou seja, responsável pela transição de estados passivo -> ativo .

Na implementação de Bryant, a estrutura que representa eventos programados tem a forma, na sintaxe de ADA:

```
type f_evento is access  futuro_evento ( pointer );

type futuro_evento is  record
                          TempoReativacao: float;
                          sinal: sincronizacao;
                          next: f_evento
                       end record;

    PrimeiroEvento : f_evento := null;
```

A variável sinal, responsável pela retenção do processo associado ao evento, é do tipo sincronização, representada pela tarefa:

```
task type sincronizacao is
  entry send;
  entry wait
end;

task body sincronizacao is
begin
  accept send;
  accept wait
end;
```

Desta forma, ao executarmos o comando sinal.wait , a tarefa sincronização irá ficar em condição de espera até que seja executado o comando sinal.send, quando, então, se realizará o "rendezvous". Podemos observar, pela estrutura do evento, que a lista de eventos é encadeada de forma simples, através do uso do ponteiro next.

O primeiro procedimento, denominado "hold" , executa os seguintes passos, na sintaxe de ADA:

```
procedure hold ( TempoAtraso: in float ) is
NovoEvento: f_evento;
begin
  -- cria o evento a ser programado e calcula o tempo de
  -- programação
  NovoEvento := new futuro_evento;
  NovoEvento.TempoReativacao := Tempo + TempoAtraso;
  -- insere o novo evento na lista de eventos
  Insere_ListaEventos(NovoEvento);
  -- acorda o processo de programação "scheduler"
  scheduler.next;
  -- atrasa o processo que invocou o procedimento hold
  -- através da chamada da tarefa sincronização
  NovoEvento.sinal.wait
end hold.
```

O processo de programação , denominado "scheduler" , executa os seguintes passos, na sintaxe de ADA:

```
task scheduler is
  entry start;
  entry next
end scheduler;

task body scheduler is
begin
  -- inicio da simulação
  accept start;
  loop
    -- a cada iteração uma tarefa é reativada
       associada ao primeiro evento da lista
    if PrimeiroEvento /= null then
      -- reativa a tarefa associada
      -- e avança o relógio
      Tempo := PrimeiroEvento.TempoReativacao;
      PrimeiroEvento.sinal.send;
    end if;
    -- aguarde a reativação
    accept next;
    -- avançe para o próximo evento
    PrimeiroEvento := PrimeiroEvento.next;
    -- verifique o término da simulação
    if PrimeiroEvento.TempoReativacao > TempoSimulacao then
      exit;
    end if;
  end loop;
end scheduler.
```

Segundo Bryant, a implementação proposta para o processo "scheduler" e para o procedimento "hold" atende as seguintes considerações :

. somente um processo é reativado a cada "loop", portanto, não existe a programação paralela de eventos;

. é observada a exclusão mútua em relação a utilização da lista de eventos;

. é observada a seção crítica dos processos (tarefas) do sistema através da condição de espera da tarefa de sincronização.

Na prática, um processo só é reativado, depois que ocorre o retorno a chamada do procedimento "hold", decorrente da liberação da tarefa de sincronização associada ao sinal do evento realizado pelo processo "scheduler". Neste sentido, sendo o procedimento "hold" reentrante, é possível a existência de vários processos aguardando a reativação ao mesmo tempo.

Basicamente, podemos traçar o seguinte diagrama, representando o mecanismo de controle proposto por Bryant, ver figura 53.

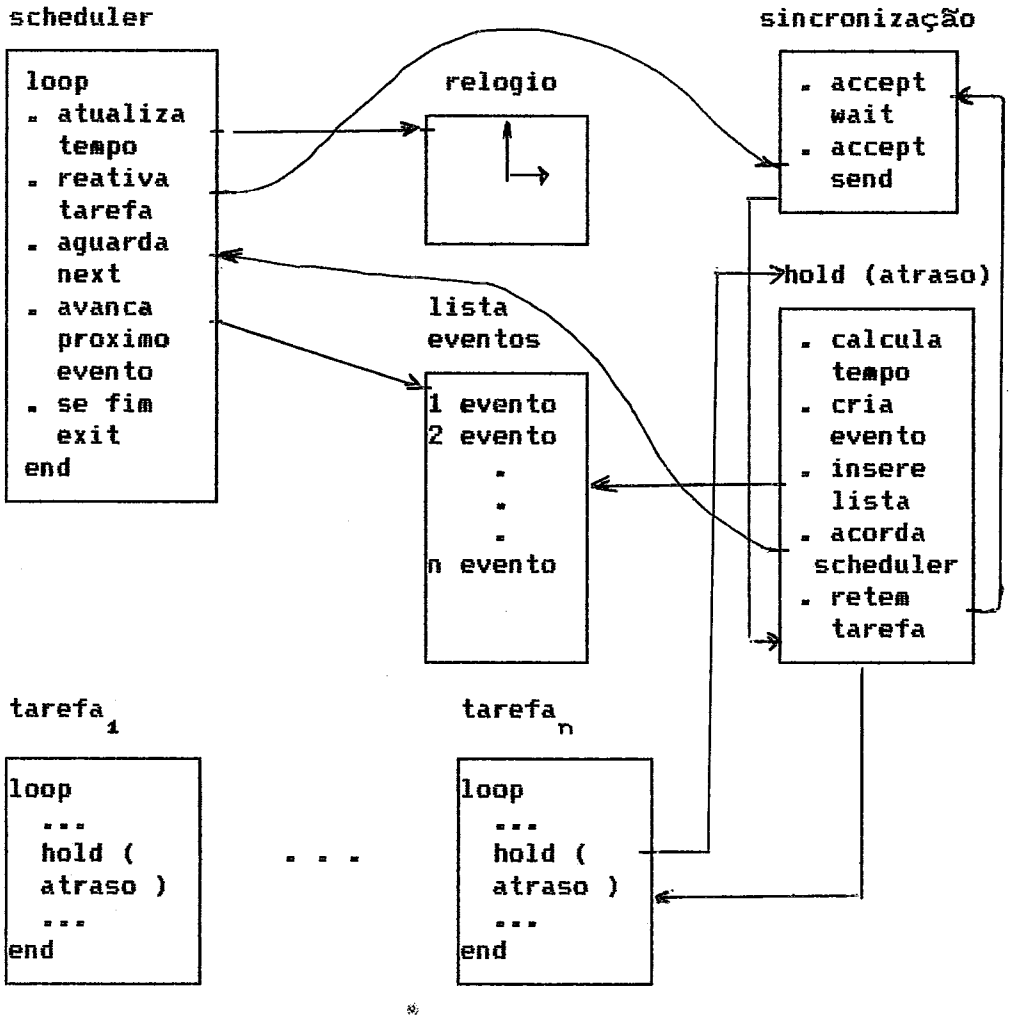


figura 53: diagrama de controle da simulação.

Neste esquema, temos quatro tipos de sincronização, em relação a exclusão mutua :

1 - Entre o procedimento "hold" e o processo "scheduler" em relação ao sequenciamento de eventos e a manipulação da lista;

2 - Entre o processo "scheduler" (através do procedimento "hold") e as tarefas do sistema em relação a reativação de tarefas;

3 - Entre as tarefas do sistema e o procedimento "hold" em relação a programação de eventos (atendendo a retrição cronológica e a observação das seções críticas), e

4 - Entre as tarefas do sistema, em relação a alguma forma de interação, que provoque a transição de uma tarefa para o estado passivo.

- Implementação com Modula-2:

Se optassemos pela implementação de Bryant na linguagem Modula-2 , teríamos que reconsiderar alguns passos do procedimento "hold" e do processo "scheduler" adaptando-os as facilidades providas pela linguagem em relação a sincronização de processos.

Verificando a existência de primitivas para a transferência de controle e criação de processos (N. Wirth [1983]) , poderíamos utilizar uma implementação de semáforos com contadores como proposta por (R.S.Wiener e G.A. Ford [1985]). Neste caso, seria possível utilizarmos as funções SEND (sinal: SIGNAL) e WAIT (sinal: SIGNAL) garantindo a sincronização entre processos.

Portanto, a cada novo evento programado, teríamos associado uma variável do tipo SIGNAL (tipo primitivo de Modula-2) relativa a um processo. Desta forma, a chamada do procedimento "hold" realizaria a suspensão do processo (trocamos a expressão NovoEvento.sinal.wait pelo comando WAIT(sinal associado ao processo)) , e o processo "scheduler" realizaria a reativação do processo (trocamos a expressão NovoEvento.sinal.send pelo comando SEND(sinal associado ao processo)) . A sincronização entre o procedimento "hold" e o processo "scheduler" se daria da mesma forma, utilizando os comandos SEND(next) e WAIT(next) no lugar das expressões scheduller.next e accept next .

R. Sharma e L.L.Rose[1988] apresentam o desenvolvimento de um pacote modular de "software" voltado para a simulação discreta de sistemas, orientado segundo a programação de eventos ou segundo a interação entre processos. Os autores, destacam, neste trabalho, o uso da linguagem de programação Modula-2, devido, principalmente, ao alto grau de modularidade suportado no desenvolvimento de sistemas de programação, a capacidade para desenvolvimento de tipos abstratos de dados, e , finalmente, a capacidade de prover mecanismos de controle entre processos baseados em co-rotinas a exemplo da linguagem Simula-67.

Para o controle da simulação, os autores desenvolveram um módulo para o gerenciamento de processos, incluindo procedimentos que permitem a criação, ativação, suspensão e término dos mesmos através do uso de primitivas da linguagem Modula-2 como o procedimento para criação de processos (NEWPROCESS) e o mecanismo para a transferência de controle entre co-rotinas (TRANSFER). Considerando a criação de processos em Modula-2 e a existência do tipo PROCEDURE, a cada processo do sistema é associado um código executável mais uma memória de trabalho, que determinam as atividades e atributos das entidades do sistema.

Os autores desenvolveram, também, um modulo que permite o gerenciamento de recursos do sistema, definindo, para tanto , procedimentos para a criação, e manipulação dos mesmos, segundo um conjunto de disciplinas de atendimento: FCFS, Prioritaria, "FirstFit", "BestFit", etc.

- Execução Concorrente de Processos:

Segundo Bryant, a simulação paralela de eventos é possível, através da reativação simultânea de vários processos associados a eventos com mesmo tempo de ocorrência. Neste caso, temos que observar a possibilidade da existência de vários processos em estado ativo ao mesmo tempo, obrigando um estudo mais aprofundado em relação a garantia de programação de futuros eventos e a verificação da exclusão mútua referente a manipulações na lista de eventos.

S. Sheppard, P. Friel e D. Reese [1984] e P. Friel e S. Sheppard [1985], apresentam o desenvolvimento de um novo sistema de simulação orientado por processos em ADA, que garante a simulação

paralela de eventos , a partir de algumas modificações introduzidas no trabalho de Bryant, de forma a garantir o sequenciamento na programação de eventos e a verificação de exclusão mútua , face a execução paralela de processos, causada pela reativação simultânea de vários eventos.

Segundo as autoras, o sistema de Bryant apresenta dois problemas em relação ao processamento paralelo de processos ou tarefas.

1 - Problema de exclusão mútua: como existem vários processos funcionando em paralelo, podemos ter , também, simultâneas tentativas de programação de eventos através do procedimento "hold" (lembrando que a chamada do procedimento é reentrante). Para tanto, é necessário garantir o acesso exclusivo a lista de eventos por parte dos vários processos (inclusive o processo "scheduler"). Isto é possível, através da transformação do módulo que implementa a lista de eventos em uma tarefa com um controle não determinístico de entrada (instrução select em ADA)

2 - Problema de sequenciamento de eventos: este problema, já abordado anteriormente, surge devido a possibilidade da programação simultânea de vários eventos. Imagine que, em certa simulação, o processo "scheduler" determine a reativação de dois processos P_1 e P_2 , associados a dois eventos programados para o mesmo instante. Em seguida, suponha que o processo P_1 determine a programação do evento e_1 para o instante t_1, que, coincidentemente, seja o tempo mínimo da lista de eventos, antes do término do processo P_2. Portanto, o evento e_1 será prontamente realizado determinando nova reativação do processo P_1.

Entretanto, caso o evento programado pelo processo P_2 (que rodava em paralelo com o processo P_1), após a reativação do processo P_1, tenha como instante um tempo inferior ao estabelecido para o evento e_1, teremos uma processamento de eventos de forma não sequencial, afetando a restrição cronológica.

Para resolver este último problema, as autoras propõem algumas modificações no sistema de Bryant, de tal forma que o processo "scheduler" observe, sempre, o número de processos ativos (processos que rodam em paralelo) e permita que algum processo seja reativado, somente quando todos os processos ativados em paralelo tenham terminado.

Segundo a nova implementação, o procedimento "hold" executa o passo de introdução do novo evento na lista de eventos de forma não determinística, através do comando:

```
-- insere o novo evento na lista de eventos de modo
-- não determinístico
ListaEventos.Insere(NovoEvento);
```

O processo de programação , denominado "scheduler" , executa os seguintes passos, na sintaxe de ADA:

```
task scheduler is
  entry start(n: in integer);
  entry next;
  entry wait;
  entry spawn;
end scheduler;

task body scheduler is
begin
  -- início da simulação
  accept start(n: in integer);
  NumeroAtivos := n;
  loop
   -- a cada iteração tarefa(s) e(são) reativada(s)
   -- associada(s) ao(s) primeiro(s) evento(s) da lista
   -- realizando-se o encontro com a tarefa ListaEventos.Remova
    index := 0;
    loop
     -- a cada iteração é aceito um sinal associado a
     -- programação (suspensão ) ou ativação de uma tarefa e
     -- também quanto a decisão de tornar uma tarefa passiva
      select
        accept next;
          index:= index + 1;
        or accept wait
          NumeroAtivos := NumeroAtivos -1;
        or accept spawn;
          NumeroAtivos := NumeroAtivos +1;
      end select;
      if index = NumeroAtivos then
        exit;
      end if;
    end loop;
    if Tempo > TempoSimulacao then
      TermineSimulacao
    else
      ListaEventos.Remova;
    end if;
  end loop;
end scheduler.
```

```
task ListaEventos is
  entry Insert ( NovoEvento : in f_evento);
  entry Remove;
end ListaEventos;


task body ListaEventos is
-- declaração de ponteiros do tipo f_evento
begin
  loop
    select
      -- inserção de eventos
      accept Insert ( NovoEvento : in f_evento ) do
      ...
      end Insert;
      -- operações do processo scheduler
      or accept Remove do
       if PrimeiroEvento /= null then
         -- avança o relógio
         Tempo := PrimeiroEvento.TempoReativação;
         NumeroAtivos := 0;
         loop
           -- reative todas as tarefas possíveis
           PrimeiroEvento.sinal.send;
           NumeroAtivos := NumeroAtivos + 1;
           -- avançe para o próximo evento
           PrimeiroEvento := PrimeiroEvento.next;
           if PrimeiroEvento.TempoReativação > Tempo then
             exit;
           end if;
         end loop;
       end if;
     end Remove;
   end select;
 end loop;
end ListaEventos.
```

No processo "scheduler" podemos observar que, através do uso do mecanismo de escolha não determinístico, o mesmo realiza o encontro e é reativado pelo sinal next proveniente do procedimento "hold". Neste instante, o número de tarefas programadas (chamadas de "hold") é acrescido de uma unidade.

Por outro lado, se o processo recebe no instante do encontro um sinal wait, significa que a tarefa associada foi colocada em estado passivo, e o processo "scheduler" não precisa mais aguardar sua programação (lembrando que a mesma só poderá ser reativada através de outra tarefa). Neste caso, o número de tarefas ativas é decrescido de uma unidade.

Finalmente, o processo "scheduler" pode receber um sinal spawn

relativo a ativação de outra tarefa por uma tarefa ativa qualquer. Agora, ao contrário do caso anterior, devemos aumentar o número de tarefas ativas em uma unidade.

A execução do "loop" seletivo só termina quando o número de tarefas ativadas for igual ao número de tarefas programadas, evitando, desta forma, o problema de não sequenciamento de eventos.

- Considerações:

A modelagem de um sistema de simulação, segundo a concepção orientada por processos, não traz grandes ganhos em relação a redução do número de eventos programados. Isto se deve, principalmente, a realização da programação sequencial de eventos e a manutenção de um relógio do sistema , como forma de garantir a integridade do sistema quanto as restrições de natureza cronológica. Neste caso, a garantia da execução de eventos , obedece ao ordenamento cronológico e sequencial de eventos ao longo do processo de simulação, ou seja, só é permitida a realização de eventos com mínimo tempo de ocorrência na lista de eventos.

Desta forma, a execução ou realização paralela de eventos só acontece, quando existe a ocorrência simultânea de eventos, o que nem sempre ocorre com muita frequência. Como podemos verificar, a execução paralela de eventos ou ativação paralela de processos, segundo a modelagem por processos, exige o estabelecimento de mecanismos de sincronização bastantes complexos, como forma de garantir a seguridade, integridade e sequenciamento da simulação, tornando esta concepção , segundo S. Sheppard, P. Friel e D. Reese [1984], menos eficiente, computacionalmente, que a concepção clássica orientada por eventos.

Porém, a grande vantagem desta concepção de simulação, talvez seja em relação a modelagem do sistema , que apesar de mais complexa, retrata com mais realidade e clareza o comportamento e funcionamento de sistemas reais de comportamento dinâmico.

6.4.3 - Dependência Lógica e Cronológica entre Eventos:

- Garantia de Realização de Eventos:

Ao que parece, a principal restrição imposta as concepções de simulação anteriores, se refere a execução sequencial de eventos segundo a ordem ou dependência cronológica.

Neste sentido, o evento com garantia de realização, é aquele com mínimo tempo de ocorrência na lista de eventos, e as propriedades de continuidade e término da simulação ficam garantidos pela programação de eventos ou ativação de processos com tempos de ocorrência que aumentam de forma gradativa até o término da simulação.

Entretanto, se obtivermos na lista de eventos uma upla (e_j,t_j ,i,s) com tempo de ocorrência t_j, que não seja o tempo mínimo, podemos afirmar que a execução deste evento é garantida se nenhum evento com tempo de ocorrência $t < t_j$, afetá-lo logicamente. Como exemplo, podemos considerar o cancelamento do evento, ou o cancelamento das decisões provocadas no conjunto de entidades s.

A condição ou não de um evento afetar lógicamente outros eventos , está relacionada a dependência lógica, que caracteriza, também, o interrelacionamento entre atividades de um processo, bem como a interação entre diferentes processos durante o funcionamento do sistema.

- Execução Paralela de Eventos sem Dependência Lógica:

D. Kumar [1986] apresenta em seu trabalho uma técnica de simulação que garante a execução paralela de eventos sem dependência lógica, atendendo as restrições de integridade do sistema; e sequenciamento, continuidade e término da simulação.

A idéia de Kumar é computar, antecipadamente, todos os eventos garantidos , ou seja, para cada entidade "i" (gera o evento) e para cada entidade "j" (afetada pelo evento) é computado um evento que é garantido de ocorrer em um instante t, criando-se uma dupla (e,t) associada a um buffer $B_{i,j}$. Em seguida, para cada entidade entidade "i" executa-se os eventos causados por ela, utilizando-se as informações do buffer $B_{i,j}$ para cada entidade "j".

E interessante observarmos que, como os eventos não são simulados na ordem cronológica , não é necessário a manutenção de um relógio do sistema, pois a dependência cronológica é reduzida a dependência lógica. Entretanto, para duas entidades "i" e "j" , todos os eventos causados pela entidade "i" que afetam a entidade "j" são realizados de forma sequencial.

Este processo de simulação reduz, a primeira vista, a inserção de eventos na lista de eventos, diminuindo, portanto, a complexidade da simulação.

Podemos verificar, também, que a complexidade das funções de manipulação dos buffers são constantes devido a indexação, enquanto que , as operações de inserção na lista de eventos , são para os melhores algorítmos, de complexidade $O(\sqrt{n})$ em relação ao número total de eventos existentes.

Por exemplo, suponha que um $trem_1$ ocupe, temporáriamente, a $seção_j$ da ferrovia, enquanto que o $trem_2$ avança na $seção_{j+2}$, conforme a situação descrita na figura 54.

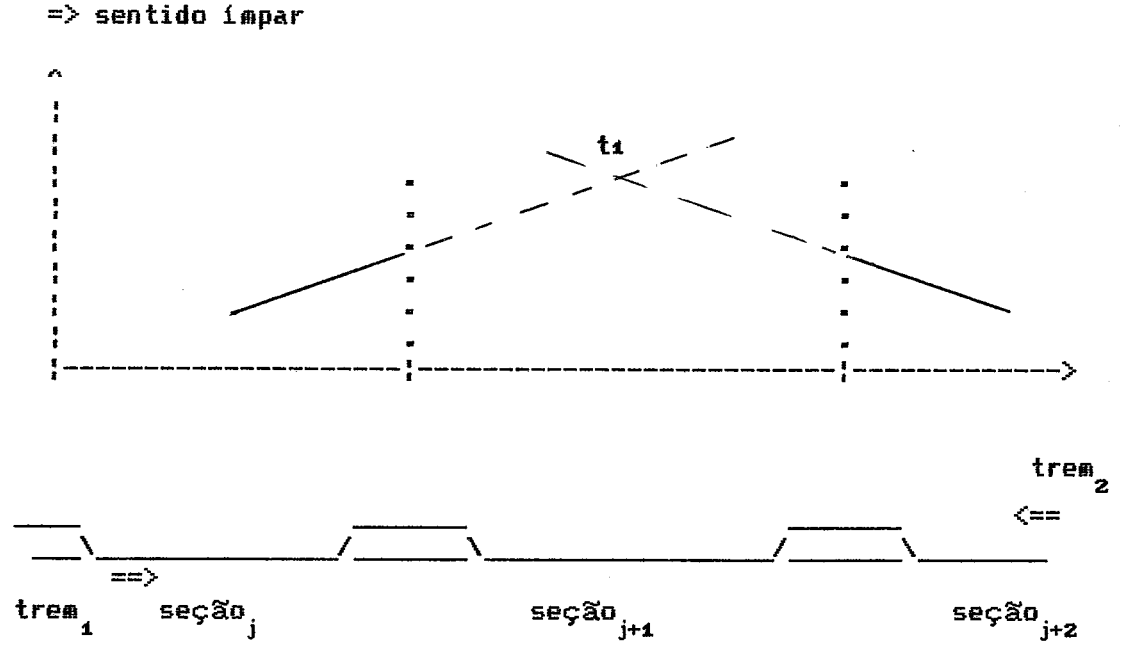


figura 54: movimento de trens.

Para representarmos a ocorrência destas atividades, temos uma lista de eventos na forma:

$(e_i, t_i$, seção "y", seção "x") , (125)

significando que um trem inicía o percurso na seção "x", oriundo da seção "y", no instante "t" , tornado o seu status ocupado, e, consequentemente, tornando o status da seção "y" desocupado.

Como podemos verificar, os eventos que correspondem as atividades de percurso do $trem_1$ na $seção_j$ e do $trem_2$ na $seção_{j+2}$ podem ser executados paralelamente, já que não existe qualquer dependência lógica entre os mesmos. Ou seja , poderiamos avançar o relógio do sistema até o término das duas atividades, visto que não ocorre a chegada de novos trens durante o processo de atendimento do $trem_1$ e do $trem_2$. Entretanto, para decidirmos a programação do evento relativo ao percurso da $seção_{j+1}$, teríamos um problema decisório associado a existência do conflito de cruzamento no instante t_1.

Este conflito, provoca a existência de um evento que determinará a forma como se processara a exclusão mútua em relação a ocupação da $seção_{j-1}$ pelos trens $trem_1$ ou $trem_2$. Observe que, se adotassemos uma disciplina de serviço do tipo FCFS (primeiro a chegar é o primeiro a ser atendido) não teríamos a existência de conflito, pois o $trem_1$ seria prontamente atendido.

Portanto, se quizermos realizar uma simulação paralela de eventos, atendento a existência de interação entre processos, teremos que observar, e , consequentemente , resolver sequencialmente os conflitos decisórios. Como pode ser observado na seção 5.5.2, existe uma dependência lógica entre estes conflitos, que, entretanto,podem ser identificados (ver seções 3.3.6 e 3.6.5) através de uma pesquisa que determina o mínimo tempo associado a um conflito em uma tabela de tempos.

- Implementação Parcialmente Correta:

Inicialmente, D.Kumar[1986], apresenta um algorítmo parcialmente correto, que realiza os seguintes passos:

```
Início
  Inicializa cada buffer;
  término <-- FALSE;
```

```
Enquanto Não(termino) Faça
  Para cada entidade "i" Faça
    Para cada buffer B(i,j) Faça
      Compute evento garantido causado por "i" e que afeta "j";
      Deposite o evento computado (e,t) no buffer B(i,j)
    Fim Para
  Fim Para;
  Se o evento (e,z) correspondente ao fim da simulação
  foi depositado Então
    término <-- TRUE
  Fim Se
 Fim Enquanto
Fim.
```

O funcionamento deste algorítmo, pode ser exemplificado pela simulação do movimento, correspondente a situação descrita na figura 55.

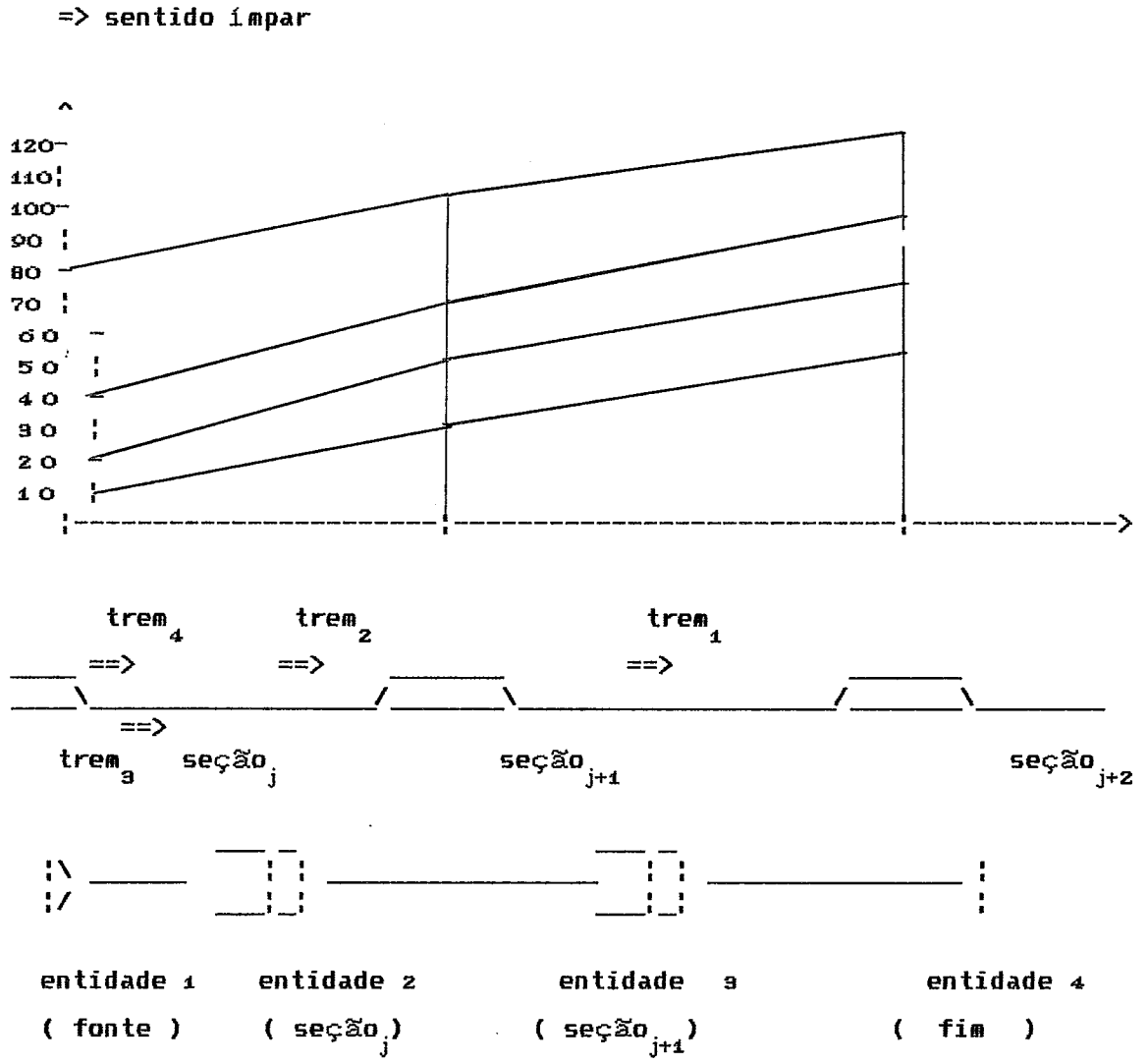


figura 55: movimento de trens segundo o algorítmo de Kumar.

Como podemos verificar, a entidade 1 está associada a uma fonte, que gera, inicialmente, os trens $trem_1$, $trem_2$, $trem_3$ e $trem_4$, vindos da $seção_{j-1}$. As entidades 2 e 3 são filas abstratas com disciplina FCFS, associadas ao serviço de atendimento de trens nas $seção_j$ e $seção_{j+1}$ e, finalmente, a entidade 4 está associada ao término do percurso dos trens.

Atendendo ao diagrama de trens da figura 55, observamos a existência de 11 eventos relacionados ao início e fim das operações ou atividades existentes nas entidades 1, 2, 3 e 4. Desta forma, ao longo do processo de simulação, teremos as seguintes situações na lista de eventos, ver quadro 6.

quadro 6: operações na lista de eventos:

lista	programação	geração	cancelamento
vazia		1	
1	1 -> 3	2	
2,3	2 -> 5	4	1
3,4,5	3 -> 6		2
4,5,6	4 -> 7		3
5,6,7	5 -> 8	9	4
6,7,8,9			5
7,8,9	7 -> 10		6
8,9,10			7
9,10	9 -> 11		8
10,11	10 -> 12		9
11,12			10
12			11
vazia			12

Para tanto, considerou-se os seguintes eventos:

1 - (trem1,10,1,{1,2}), ou seja, o $trem_1$, gerado pela entidade 1, inicía a operação na entidade 2, no instante 10.

2 - (trem2,20,1,{1,2})
3 - (trem1,30,2,{2,3})
4 - (trem3,40,1,{1,2})
5 - (trem2,50,2,{2,3})
6 - (trem1,50,3,{3,4})
7 - (trem3,70,2,{2,3})
8 - (trem2,70,3,{3,4})
9 - (trem4,80,1,{1,2})
10 - (trem3,90,3,{3,4})
11 - (trem4,100,2,{2,3})
12 - (trem4,120,3,{3,4})

Aplicando o algorítmo de Kumar, teríamos o cômputo das seguintes uplas e , consequentemente, o depósito, nos buffers apropriados:

Para a entidade 1

Para o buffer $B_{1,2}$

depósito do evento (trem1,10,1,2)
depósito do evento (trem2,20,1,2)
depósito do evento (trem3,40,1,2)
depósito do evento (trem4,80,1,2)

Para a entidade 2

Para o buffer $B_{2,3}$

depósito do evento (trem1,30,2,3)
depósito do evento (trem2,50,2,3)
depósito do evento (trem3,70,2,3)
depósito do evento (trem4,100,2,3)

Para a entidade 3

Para o buffer $B_{3,4}$

depósito do evento (trem1,50,3,4)
depósito do evento (trem2,70,3,4)
depósito do evento (trem3,90,3,4)
depósito do evento (trem4,120,3,4)

Como podemos verificar, na simulação do sistema caracterizado pela figura 55, não houve necessidade de inserções na lista de eventos. Também, como , para cada buffer $B_{i,j}$ a sequência de eventos causadas por "i" e que afetam "j" consiste de uma subsequência ordenada de eventos garantidos, segundo a ordem cronológica, fica garantido a propriedades de sequenciamento e de integridade do sistema.

- Existência de "Loop" Infinito:

Entretanto, o algorítmo descrito por Kumar pode, as vezes, cair em "loop" infinito. Ou seja, em determinado momento, não se pode assegurar a garantia de algum evento. Portanto, o algorítmo apresentado anteriormente fica parcialmente correto, já que não da garantias da continuidade e término da simulação. Seja o exemplo representado na figura 56.

Considerando:

fontes: entidades 1, 2 e 3;

serviços de atendimento: entidades 5 e 7;

serviços de recepção: entidades 4,6 associadas as entidades 5,7;

fim: entidades 8 e 9.

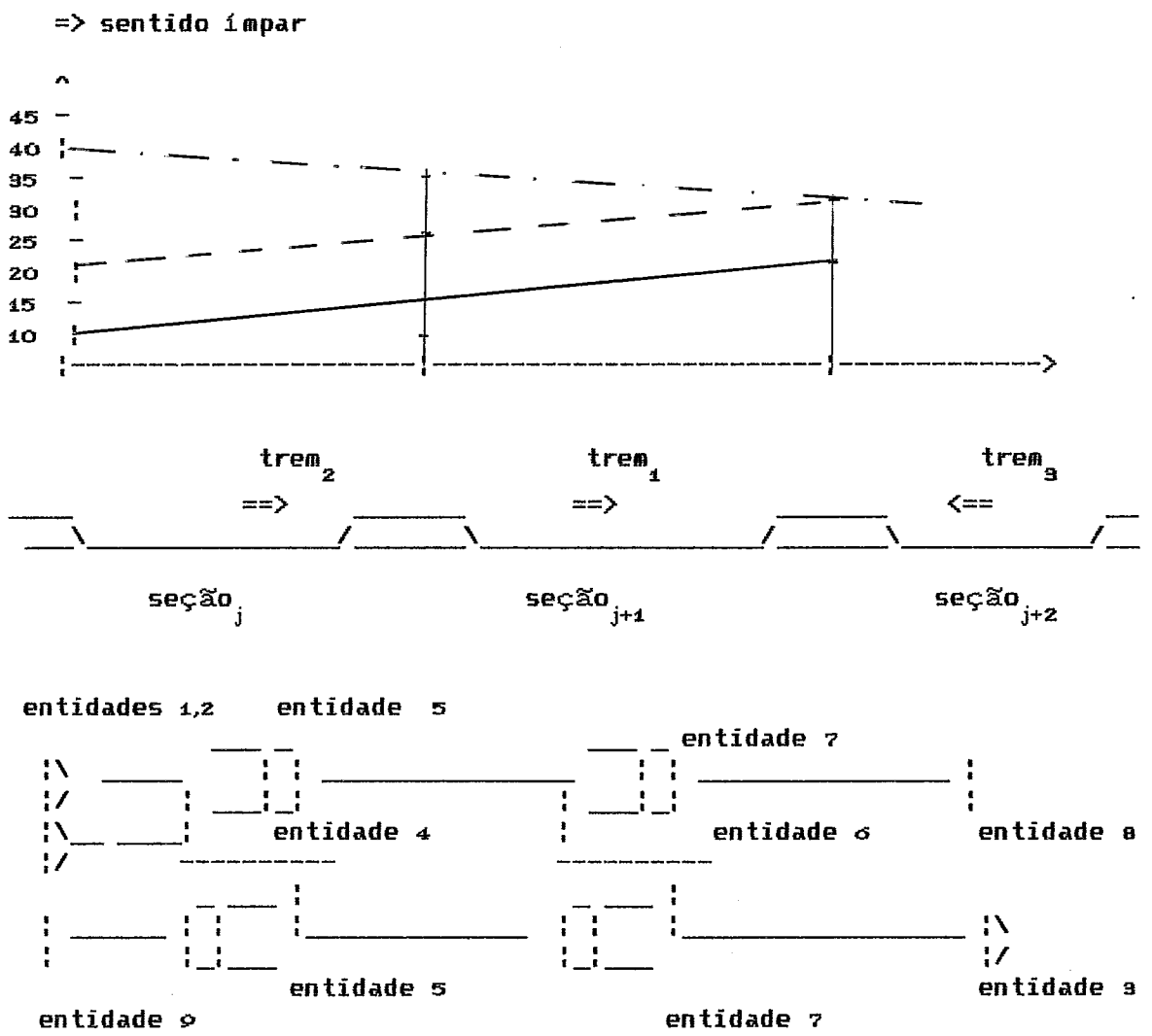


figura 56: movimento de trens com ocorrência de "loop" infinito.

Eventos gerados inicialmente:

(trem1,10,1,{1,4})
(trem2,20,2,{2,4})
(trem3,30,3,{3,6})

Aplicando o algorítmo de Kumar, teriamos o cômputo das seguintes uplas e , consequentemente, o depósito, nos buffers apropriados:

Para a entidade 1
  Para o buffer $B_{1,4}$
    depósito do evento (trem1,10,1,4)

Para a entidade 2
  Para o buffer $B_{2,4}$
    depósito do evento (trem2,20,2,4)

Para a entidade 3
  Para o buffer $B_{3,6}$
    depósito do evento (trem3,30,3,6)

Para a entidade 4
  Para o buffer $B_{4,5}$
    depósito do evento (trem1,10,4,5)

Para a entidade 5
  Para o buffer $B_{5,6}$
    depósito do evento (trem1,15,5,6)

Para a entidade 6
  Para o buffer $B_{6,7}$
    depósito do evento (trem1,15,6,7)

Para a entidade 7
  Para o buffer $B_{7,8}$
    depósito do evento (trem1,20,7,8)

Este último evento, só é garantido devido a existência da disciplina de atendimento do tipo FCFS.

Neste ponto, a entidade 4 não pode garantir a execução do evento (trem2,20), ou seja , não pode depositar a upla (trem2,20,4,5) no buffer $B_{4,5}$ já que a entidade 4 não sabe se outra upla , gerada, por exemplo, a partir da entidade 1 será, também, depositada . Similarmente, a entidade 6 também não pode depositar nenhuma upla no buffer $B_{6,7}$.

Portanto, o algorítmo entra em "loop" infinito antes de completar as atividades dos trens $trem_2$ e $trem_9$.

- Implementação Totalmente Correta:

A solução proposta por Kumar, consiste em associar ao algorítmo anterior, uma lista de eventos que contém sempre os próximos eventos a serem executados segundo a ordem cronológica de ocorrência . Desta forma, quando o algorítmo detectar um "loop" ou a inexistência de eventos garantidos, durante o exame das entidades, o mesmo recorre a lista de eventos escolhendo aquele com mínimo tempo de ocorrência.

Neste instante, o evento garantido passa a ser aquele com tempo mínimo na lista de eventos. Escolhido o evento , o mesmo é depositado no buffer apropriado e o algorítmo tenta, novamente, computar os eventos garantidos atraves do exame das entidades. Desta forma, a cada situação

de loop, o algorítmo retorna a lista de eventos.

Portanto, o algorítmo totalmente correto, possue os seguintes passos:

```
Início
  Inicializa cada buffer;
  término <-- FALSE;
  Enquanto Não(término) Faça
    Repita
      Para cada entidade "i" Faça
        Para cada buffer B(i,j) Faça
          Compute evento garantido causado por "i" que afeta "j";
          Deposite o evento computado (e,t) no buffer B(i,j) ;
          Evento_Computado <- TRUE ;
        Fim Para
      Fim Para;
      Se não há mais evento computável Então
        Evento_Computado <- FALSE
      Fim Se
    Ate Não(Evento_Computado);
    Para cada entidade "i" Faça
      Construa lista de eventos, computando eventos programados;
    Fim Para;
    Se a lista de eventos está vazia Então
      término <- TRUE
    Senão
      Escolha evento (e,t,i,s) com mínimo tempo de ocorrência e
      deposite a upla (e,t) nos buffers B(i,j) para todo j ∈ s;
    Fim Se
  Fim Enquanto
Fim.
```

Desta forma, como no algorítmo anterior, são computados, inicialmente, todos os eventos garantidos através do exame das entidades.

Quando não houver mais eventos garantidos, o algorítmo recorre ao evento programado com tempo mínimo na lista de eventos. Neste sentido, podemos observar que, na hipótese de nunca haver eventos garantidos na primeira fase do algorítmo, o mesmo comporta-se como o algorítmo orientado pela programação de eventos. Neste caso, a dependência lógica entre eventos é igual a dependência cronológica e o evento garantido será, sempre, aquele com tempo mínimo na lista de eventos.

Entretanto, se não houver dependência lógica entre eventos, causada pela interação de processos, o algorítmo terminará sem a necessidade de se construir a lista de eventos, como no exemplo da figura 55.

No exemplo anterior, da figura 56, teríamos, apos a ocorrência do "loop" a construção de uma lista de eventos, com os eventos programados: (trem2,20,4,{4,5}) e (trem3,30,6,{6,7)}. Neste ponto, o algorítmo escolhe o evento (trem2,20,4,{4,5}) que determina o depósito do evento (trem2,20) no buffer $B_{4,5}$, gerando em seguida, o depósito das uplas:

```
Para a entidade 5
  Para o buffer B5,6
    depósito do evento (trem2,25,5,6)
Para a entidade 6
  Para o buffer B6,7
    depósito do evento (trem2,25,6,7)
Para a entidade 7
  Para o buffer B7,8
    depósito do evento (trem2,30,7,8)
```

Novamente, é detectada uma situação de "loop" infinito e a lista de eventos é construída, possuindo o evento: (trem3,30,6,{6,7}). Neste ponto, o algorítmo escolhe este evento e determina o depósito do evento (trem3,30) no buffer $B_{6,7}$, gerando em seguida, o depósito das uplas:

```
Para a entidade 7
  Para o buffer B7,4
    depósito do evento (trem3,35,7,4)
Para a entidade 4
  Para o buffer B4,5
    depósito do evento (trem3,35,4,5)
Para a entidade 5
  Para o buffer B5,9
    depósito do evento (trem3,40,5,9)
```

Em seguida, é detectada, novamente, uma situação de "loop" infinito , mas a simulação termina, pelo fato de não haver mais eventos programados .

- Considerações:

Em particular, queremos atribuir duas críticas ao sistema de simulação apresentado por Kumar:

1 - Não foi levantada a possibilidade de uma disciplina de atendimento que leve a um conflito de decisão, no que tange a utilização simultânea de recursos. No exemplo anterior, vimos que a utilização de uma disciplina do tipo FCFS permitiu a garantia do evento (depósito no buffer). Entretanto, se adotassemos outra disciplina de atendimento, não teriamos a garantia do evento. Mais ainda, teríamos a geração de um novo evento associado ao conflito decisório, cuja realização, afetaria, lógicamente, a ocupação da entidade ou a ordem de processamento. Na teoria de "scheduling", estamos nos referindo ao cumprimento das restrições de precedência, que garantem a exclusão mútua em relação aos recursos do sistema.

2 - Nao foi levantada a possibilidade da realização de um evento garantido provocar o cancelamento e consequente reprogramação de outros eventos, durante o processo de simulação. Novamente, no exemplo anterior, se o serviço de atendimento na entidade 7 fosse de 10 unidades para o processamento do $trem_2$, determinando o depósito da upla (trem2,35,7,5), teríamos o cancelamento do evento (trem3,30,6,{6,7}) e a programação do evento (trem3,35,6,{6,7}).

A capacidade de programar eventos e , posteriormente, realizar operações de cancelamento de modo eficiente, e uma característica, raramente oferecida pelas linguagens de simulação. Entretanto, esta característica é de fundamental importância na simulação de sistemas de planejamento e "scheduling" , onde o término das operações que determinam o tempo de ocorrência dos eventos é modificado constantemente , segundo vários tipos de interferência.

6.4.4 - Modelo de Simulação associado ao Sistema SIGT

- Mecanismos de Controle e Programação:

Em nosso sistema de simulação consideramos, inicialmente, o

cômputo de todos os eventos relacionados as atividades de atendimento e chegada (percurso dos trens nas seções) atraves da geração prévia de todos os eventos programados (geração de trens), bem como das variáveis aleatórias associadas aos tempos de atendimento (tempos de percurso) .

Desta forma, construímos uma tabela de tempos que representa a realização de todos os eventos associados as restrições de sequenciamento do problema. Esta tabela, apresentada nas seções 3.6.1 e 3.6.4, , atende , a primeira vista, as condições de sequenciamento, continuidade e término da simulação. Entretanto, a mesma não satisfaz as condições de integridade do sistema , associadas as retrições de precedência do problema, as quais exigem um mecanismo de exclusão em relação a disputa de recursos.

Portanto, torna-se necessário realizar um exame das atividades em cada entidade do sistema, de modo a identificar possíveis conflitos que causem violações na integridade do mesmo, conforme observações feitas anteriormente. Estes conflitos , associados a eventos decisórios, afetam lógicamente todos os eventos sucessivos que sejam influenciados pelo tipo de decisão. Desta forma, para uma dada instância da tabela de tempos, podemos considerar como eventos garantidos todos aqueles que possuem tempos de ocorrência inferiores ao tempo mínimo de ocorrência do próşimo conflito decisório.

Sendo assim, o algorítmo proposto, apos a realização de todos os eventos associados as restrições de sequenciamento, detecta, a cada iteração, o próximo evento garantido, equivalente ao conflito decisório com tempo minimo de ocorrência , através de uma pesquisa na tabela de tempos.

A realização deste conflito, consiste em atribuir uma preferência na utilização do recurso associado , propiciando, desta forma, a exclusão mútua em relação a disputa existente. Em seguida, o algorítmo realiza o cancelamento e a reprogramação dos eventos gerados , afetados lógicamente pela tomada de decisão, atráves de uma atualização linear de uma linha da tabela de tempos.

E evidente, que o algorítmo termina, quando todos os conflitos do tipo decisório forem resolvidos para o tempo de simulação estabelecido, garantindo, neste caso, a integridade do sistema, através

da satisfação das restrições de precedência do problema.

Neste ponto, queríamos fazer algumas observações a respeito da estratégia de controle e programação por nos sugerida, tecendo algumas comparações em relação as técnicas de simulação apresentadas anteriormente:

1 - A tabela de tempos representa situações parciais ou instâncias particulares da lista de eventos em relação aos eventos programados mais o cômputo de eventos garantidos;

2 - A pesquisa do próximo evento garantido na tabela de tempos, equivalente ao evento do tipo decisório com mínimo tempo de ocorrência, é semelhante ao processo de construção e pesquisa da lista de eventos considerado no algorítmo de Kumar. Entretanto, conforme análise anterior , uma sequência de eventos do tipo decisório ao longo do tempo, não esta, necessariamente, em ordem cronológica , mas sim em ordem lógica. Esta observação, não afeta as propriedades de continuidade e término da simulação , uma vez que o número de conflitos do tipo decisório é finito e computável em função da entrada do problema;

3 - A exemplo do algorítmo de Kumar, poderíamos construir , sucessivamente, a simulação do sistema, através do depósito e realização de eventos garantidos. Ou seja, seriam realizados somente os eventos com tempo de ocorrência inferiores ao tempo mínimo de ocorrência do próximo conflito do tipo decisório. Ao terminar a realização destes eventos , seria realizado o próximo conflito decisório, permitindo, desta forma a relização de novos eventos garantidos em função da decisão efetuada. O processo se repetiria até o término da simulação. Entretanto, a utilização deste processo seria, em alguns pontos, deficiente, pois:

. o cálculo do instante do próximo conflito do tipo decisório é realizado através de um método projetivo, ver seção 3.6.5 ,que depende do tempo de ocorrência de futuros eventos programados, ainda não garantidos. Como pudemos verificar anteriormente, a pesquisa do próximo conflito do tipo decisório, fica bastante facilitada pelo uso da tabela de tempos;

. ao utilizarmos a tabela de tempos, representamos não só , uma

visão parcial dos eventos garantidos e programados, como também, o efeito provocado pela realização dos eventos até então. Como vimos, na seção 3.6.3 , é possível em função dos valores proporcionados pela tabela de tempos, efetuarmos medidas quantitativas, como exemplo: o cômputo de uma função de avaliação ou heurística, que possibilitam a orientação das decisões ao longo da simulação;

. o uso da tabela de tempos evita os procedimentos de inserção e deleção de eventos, característicos da simulação orientada por eventos. Ademais, as operações de atualização (cancelamento e programação de eventos) bem como todas as operações de acesso, são de tempo constante , devido ao fato da estrutura representativa dos eventos, ser uma estrutura estática;

4 - Finalmente, queremos ressaltar, que através do uso de múltiplas tabelas de tempos (cada uma representando uma simulação particular), é possível estabelecermos métodos de otimização. Nos sistemas de simulação que mantem uma lista de eventos, esta possibilidade não chega sequer,a ser cogitada, dada a grande dificuldade de se manter, ao mesmo tempo, múltiplas listas de eventos, necessárias a representação simultânea de diferentes linhas de ação.

- Trabalhos Correlatos:

O Sistema para a produção de "schedules" desenvolvido por G. Bruno, A. Elia e P. Laface [1986] empregam uma forma semelhante de exame de atividades de modo a permitir o avanço da simulação. Esta técnica de simulação é considerada , em geral, ineficiente, dada a necessidade de se examinar, repetidamente, as condições lógicas que determinam a execução das atividades em cada entidade do sistema a cada passo da simulação. Entretanto, esta conclusão não se aplica a simulação de sistemas de produção de "schedules" e, especialmente, ao problema de sequenciamento de trens, como vimos anteriormente.

A diferenca básica existente entre o nosso sistema de simulação e o sistema apresentado por Bruno, Elia e Laface , é que, a programação de eventos neste último sistema, é feita, segundo a ordem cronológica, tomando-se, sempre, o evento com mínimo tempo de ocorrência, após o exame de atividades, e cancelando-se os demais. Desta forma, não é

sugerida a execução paralela de eventos sem dependência lógica segundo o trabalho de D. Kumar[1986].

O primeiro trabalho que discutiu a questão de ordenamento de eventos em um sistema distribuido, formado por um conjunto de processos físicamente separados, se deve a L. Lamport [1978]. Neste trabalho, Lamport propõe a introdução de um conjunto de relógios lógicos associados a cada processo do sistema, observando as seguintes condições basicas:

" Para dois evento "a" e "b", se o evento "a "acontece antes do evento "b", então a função de tempo associada ao evento "a" é inferior a função de tempo associada ao evento "b" " .

Ou seja:

Se a --> b Então C(a) < C(b), (127)

onde C(.) se refere a função de tempo associada ao relógio de um processo que contém o respectivo evento, sendo, também, uma forma de se associar um valor número a ocorrência do evento.

Observando esta condição, Lamport apresenta duas condições, que controlam a relação de tempo em relógios associados aos processos, segundo a interdependência entre eventos, ou seja:

1 - Se "a" e "b" são eventos pertencentes a um mesmo processo "i", e o evento "a" acontece antes do evento "b", então $C_i(a) < C_i(b)$;

2 - Se "a" e "b" são eventos de processos distintos, "i" e "j" , e a ocorrência de "b" depende da ocorrência de "a", então $C_j(b) > C_i(a)$.

Portanto, para se garantir a integridade e sequenciamento do sistema de relógios devemos, para cada processo, observar as condições de avanço do tempo ou incremento da função de tempo , estabelecidas por Lamport , as quais atemdem as condições estabelecidas anteriormente:

1 - Cada processo "i" incrementa C_i(seu relógio) entre dois eventos sucessivos;

2 - Se a ocorrência de um evento "b", em um processo "j" está condicionado ao envio de uma mensagem pelo processo "i" (evento "a") associada ao instante de tempo "t", ou seja , $C_i(a) = t$, então devemos fazer $C_j \geq C_j(b) > t$.

Neste sentido, podemos considerar o sistema de relógios lógicos de Lamport, como uma justificativa para o mecanismo de avanço de tempo , , que corresponde as atualizações da tabela de tempos, em nosso sistema de simulação.

Ou seja, na existência de conflitos, atualizamos, sucessivamente, o tempo dos relógios, considerando o tempo de ocorrência de todos os eventos associados as restrições de sequenciamento do problema, ou eventos consecutivos de mesmos processos. Entretanto, na presença de um conflito, atualizamos o valor de um relógio (início de operação de um trem) em função do término da operação do trem relacionado ao segundo processo, garantindo, desta forma, as restrições de precedência do problema.

- Tomada de Decisões e Otimização:

A correspondência entre a tabela de tempos e a lista de eventos, permite a realização de uma busca em profundidade direcionada pela solução de conflitos ou pela realização de eventos decisórios. Se mantivermos, a medida em que avançamos a simulação, as tabelas de tempos organizadas em uma lista encadeada , podemos realizar, a qualquer momento, o retorno a tabela imediatamente anterior, o que representa o retrocesso da simulação ao tempo de ocorrência do último conflito decisório.

Desta forma, podemos, ao longo da simulação, avançar e retroceder a qualquer instante associado a ocorrência de um conflito do tipo decisório, permitindo uma forma de avaliação e, consequentemente, uma escolha efetiva na tomada de decisões. Neste particular, queremos ressaltar que a tabela de tempos representa não só uma instância particular da lista de eventos, mas também, contém todas as informações relacionadas a modificação dos atributos das entidades. Portanto, ao retrocedermos a tabela de tempos anterior, temos, automáticamente, todas as informações que permitem caracterizar um novo instante do processo de simulação.

Por outro lado, se realizarmos uma busca em largura, ou mesmo uma busca em profundidade com "bactracking", associada a alguma medida de avaliação discutida anteriormente, é possível selecionarmos a melhor simulação refletindo a variação quanto a possibilidade de diferentes decisões.

O processo básico de controle da simulação em nosso sistema, consiste de um processso seletivo que, a cada passo, avalia o efeito da tomada de decisão , proporcionando ao operador, a escolha do caminho de simulação mais adequado. Ou seja, é permitido ao operador avaliar o efeito da tomada de decisão e , consequentemente, selecionar, de alguma forma, a disciplina de atendimento associada a utilização das entidades (seções) do sistema. Como exemplo, podemos considerar que o operador poderia optar pelo atendimento, em primeiro lugar, ao trem que chegasse primeiro, ou , de modo divergente, ao trem de maior prioridade, proporcionando, desta forma, a escolha de diversas opções segundo vários critérios de decisão.

A escolha do caminho mais adequado, pode ser feita através do retorno a estados anteriores da simulação, o que possibilita um forma de comparação de alternativas. A possibilidade de mantermos múltiplas representações da tabela de tempos permite ao operador realizar uma minuciosa análise da história da simulação, possibilitando, a qualquer momento, o retorno ao passado, o reexame de situações e , consequentemente, a escolha de novos caminhos que conduzam, posteriormente, a novos cenários de mudanças.

Finalmente, queremos destacar que a técnica de simulação apresentada por nós, garante a simulação paralela e instantânea de um conjunto de eventos garantidos lógicamente, os quais caracterizam os processos de atendimento e chegada. Neste aspecto, podemos afirmar que a técnica proposta é perfeita quando comparada a operação real do sistema. Ou seja, admitindo a presença de um único controlador, o grau de paralelismo apresentado pelo sistema de simulação é exatamente igual ao processo real de controle e gerenciamento de trens, considerando que , neste caso, os conflitos, ou eventos decisórios, são , igualmente, resolvidos de forma sequencial.

O primeiro trabalho que estabeleceu algum tipo de

interrelacionamento entre modelos de simulação e estratégias de solução de problemas, como por exemplo, a realização de um processo de busca em um espaço de estados, foi descrito por W. Kreutzer [1979]. Neste trabalho, Kreutzer , enfatiza que um modelo de simulação representa uma única possível representação de um dado espaço de estados associada a um processo de caminhamento que leva, a partir de um estado inicial a um estado final desejável. Neste sentido, conforme nossas observações, operadores, estratégias de seleção de operadores e funções de avaliação de estados podem serem agregados ao modelo de modo a possibilitar um controle mais eficiente do processo de simulação.

- Integração com Sistema Especialista:

A realização das atividades de atendimento por uma entidade do sistema, está condicionada a satisfação de um conjunto de restrições operacionais e físicas do mesmo, que podem estar associadas ao exame de condições lógicas geradas por eventos dependentes, nos sistemas de simulação tradicionais.

Para realizarmos, sucessivamente, o exame destas condições, invocamos o uso de um sistema de produção que verifica a viabilidade destas restrições específicas e determinam preferências, de modo a orientar o operador quanto a tomada de decisões. Neste sentido, conforme já citamos, temos uma combinação na qual o modelo de simulação interage com um sistema especialista.

Conforme observações de Robert O'Keef [1986], isto é natural quando o modelo de simulação é desenvolvido para um sistema complexo, onde o sistema de produção já existe como parte integrante do processo decisório dentro do sistema. Entretanto, conforme considerações feitas na seção 2.2.3 , o sistema SIGT apresenta outras formas de combinação, ou seja, podemos aferir o sistema de produção e a base de conhecimento pelo uso do modelo de simulação, como , também, proporcionar a avaliação e geração de novas produções, através do exame detalhado de situações (ver seção 5.5).

A verificação da viabilidade do sistema SIGT , quanto a capacidade de planejamento e replanejamento em tempo real, é realizada por um modelo de monitoração. Este modelo, que sera discutido com mais

detalhes no capitulo posterior, possibilita uma real simulação da imprecisão do sistema, tendo como orientação básica, um plano de ações estabelecido préviamente, através da realização de um planejamento. Portanto, o sistema SIGT permite ao operador não só monitorar um modelo de simulação, como, também, simular a monitoração de planos.

Quanto a eficiência do sistema de simulação, quando comparado aos sistemas de simulação tradicionais, podemos considerar que, para a realização de uma corrida de simulação ótima, com a elaboração de um plano ótimo, o sistema gera um número de estados (configurações da tabela de tempos) inferior ao número de inserções na lista cronológica de eventos, que são necessárias para a realização de uma única corrida de simulação, nos modelos tradicionais.

Capítulo VII
Planejamento e Execução de Planos

Apesar de o sistema SIGT ter sido apresentado, até então, como uma arquitetura voltada para a solução de problemas relacionados ao gerenciamento e controle do tráfego de trens, achamos por bem incluir o desenvolvimento de um capítulo que trata de questões relacionadas com a teoria de planejamento, haja visto que, dentre as principais tarefas executas por nosso sistema, destacam-se a realização de planejamento, e , também, a execução e monitoração de planos.

Neste capítulo, em primeiro lugar, apresentaremos a evolução da teoria de planejamento destacando os principais sistemas existentes, segundo a forma de representação de planos e controle das ações. Em seguida, trataremos de questões relacionadas ao planejamento temporal e planejamento com incerteza Mais adiante, daremos ênfase a questão de execução e monitoração de planos. Neste sentido, destacaremos o estabelecimento de ações para o gerenciamento do tráfego de trens, ressaltando o aspecto dinâmico envolvido neste tipo de planejamento .

Abordaremos, também, o problema de reelaboração de planos , definindo as principais estratégias de reparo e as condições necessárias para se estabelecer o replanejamento, incluindo: a reavaliação de parâmetros, redefinição de objetivos e integração de planos consecutivos e adjacentes.

Por fim, apresentaremos o desenvolvimento de um simulador de planejamento que nos permite realizar a execução e monitoração de planos segundo o processo real de tomada de decisões realizado pelo controlador a partir de um centro de controle remoto, simulando o movimento de trens com a introdução de um parâmetro que regula a imprecisão do sistema.

7.1 - Introdução ao Gerenciamento de Planos de Ações

7.1.1 - Fases de um Plano

Em nosso trabalho, consideramos a tarefa de planejamento como a tarefa de elaboração de um conjunto de ações ou plano, de modo que , quando executadas, permitam a realização do controle e gerenciamento de

um sistema em tempo real, satisfazendo um conjunto de objetivos e restrições. Para tanto, podemos considerar a divisão da tarefa de planejamento em dois processos fundamentais: elaboração (ou geração e seleção) e coordenação (interpretação, execução e monitoração) . E. Charniak e D. McDermott [1985] consideram a atividade de planejamento compreendendo as seguintes etapas (ver seção 2.1.3) : planejamento tático ou projeção, interpretação e execução, monitoração e replanejamento.

As fases de planejamento e replanejamento se referem, claramente, ao processo de elaboração, enquanto que a interpretação, execução e monitorção se associam ao processo de coordenação.

A respeito da interpretação de planos, podemos considerar a manutenção de uma lista ou conjunto de ações como tal. Ou seja, do produto do planejamento, extraímos uma estrutura declarativa, parcialmente ordenada, da qual é possível obtermos , diretamente, as ações e todas as informações necessárias a perfeita execução do planejamento .

7.1.2 - Representação de Informações

Para a realização do planejamento é necessário um conjunto básico de conhecimentos os quais possibilitam desenvolver um programa ou um planejador que seja capaz de produzir planos corretos e confiáveis.

Como visto anteriormente, no capitulo III, optamos por uma abstração matemática do problema inserido na tarefa de planejamento. A extensão desta abstração permitiu-nos desenvolver um sistema físico simbólico associado a representação espaço-estado do problema, incorporando um conjunto de operadores que produzem o mesmo efeito de uma ação , ou seja, provocam uma transição legal entre estados.

Segundo E. A. Rich [1983], sendo possível a completa representação de sucessivos estados durante o processo de solução do problema, não se torna necessário o desenvolvimento de um sistema cáassico de planejamento que estabeleça a representação correta das mudanças verificadas em função do emprego de ações que provocam alteração de estados ("frame problem" (J. McCarthy e P.J. Hayes [1969])). No sistema SIGT, dado o desenvolvimento de uma estrutura de

representação de estados eficiente, ver seção 3.6.1, foi possível mantermos as informações durante todo o processo de elaboração de planos de forma a refletir as sucessivas transições.

Segundo T. Linden[1988] para possibilitar a execução de planos necessitamos de:

1 - Uma lista de ações parcialmente ordenada : esta estrutura constituí uma forma simples e eficiente de representação de ações. No sistema de controle e gerenciamento de trens esta lista deve conter as informações necessárias a identificação e solução de conflitos;

2 - Informações ou restrições temporais: são informações que permitem o estabelecimento de um ordenamento entre ações. Ou seja, o instante ou intervalo de tempo na qual a execução das ações devem ocorrer, bem como as relações de precedência envolvidas. Em nosso sistema, fazemos uso de lista de ações sequenciais, ordenadas segundo o tempo de ocorrência dos conflitos de acordo com as projeções conferidas;

3 - Parâmetros formais: que possibilitarão o casamento de ações com objetos reais resultando na ligação de planos. Neste caso, se a ligação acontecer, teremos a viabilidade da ação ou sucesso parcial do plano. Caso contrário, dissemos que o plano falhou ou que a ação e inviável ou inconsistente com a situação real. A respeito da falha de planos, quando o casamento da ação falha , podemos ter a perda ou falha total do plano, exigindo, imediamente , a realização de replanejamento. Entretanto, em alguns casos, é possível o reparo do plano, evitando, desta forma, a perda ou falha total;

4 - Uma lista de pré-condições: em alguns modelos de planejamento, utiliza-se uma lista de pre-condições, separadas do corpo da ação, de modo que suas informações devam ser conferidas a fim de garantir a execução das ações.

Em adição a estes itens principais, podemos considerar tambem:

1 - O efeito das ações: ou seja, a caracterização física da transição entre estados ;

2 - Uma medida de alcance dos objetivos : o que é essencial para se avaliar o desempenho dos planos em relação ao cumprimento dos

objetivos, e, finalmente ,

3 - Informações dependentes: estas informações são relativas ao efeito das ações associadas ao nível de alcançe ou satisfação dos objetivos. Em nosso sistema, são de fundamental importância para o processo de replanejamento pois permitem efetuarmos a reavaliação de prioridades, redefinição de objetivos e recomposição do espaço de estados do problema.

7.1.3 - Caracterização de Planos

No gerenciamento de plano de ações, podemos caracterizar um conjunto de elementos básicos que serão importantes na discussão da teoria de planejamento , envolvendo uma teoria de tempo e ação, que estuda a dimensão temporal do planejamento. Segundo P.R. Young e P.E. Lehner [1986] a segunda dimensão do planejamento está relacionada a hierarquia ou níveis de abstração.

Para tanto, consideremos o desenvolvimento existente no trabalho de A. Haas [1985] que utiliza uma linguagem de planos e uma notação semelhante a empregada pelo sistema de planejamento STRIPS (R.E. Fikes e N.J. Nilsson [1971]). No seu trabalho, Haas introduz uma teoria de intervalos de tempos, ao contrário do sistema STRIPS que utiliza um tipo de calculo de situações para raciocinar sobre possíveis ações e seus efeitos.

Para Haas uma situação consiste de uma descrição completa de um estado constante do mundo ou domínio do planejamento. Ou seja, se "s" e uma situação e "i" é um intervalo de tempo, então o mundo permanece na situação "s" durante o intervalo "i". Uma troca de situações é proporcionada por uma ação que provoca a criação de uma nova situação , digamos "s'", ou a transformação s --> s'. Frequentemente, nos modelos clássicos de planejamento é utilizado a notação de estado para caracterizar situações.

Se um agente, que provoca a ação, tem completo conhecimento da situação atual, e , também , do efeitos da ação, podemos caracterizar um plano como uma sequência de ações executáveis. Entretanto, na impossibilidade de se executar alguma ação, temos uma interrupção do

plano. Para resolver este problema , Haas introduz a idéia de comando. Ou seja, um agente possui um plano ou programa de ações e uma rotina de execução. Em determinado momento, o agente seleciona uma ação apropriada e envia a rotina de execução um comando. A rotina de execução, por sua vez, toma a ação desejável.

Desta forma, uma ação é ou não executada, dependendo da situação, mas o agente e o processo de planejamento não se interrompem. Portanto, nesta nova filosofia, podemos considerar um plano como uma sequência de comandos que permite ou não a sequência de ações serem executadas. Lógicamente, se um ação não puder ser executada, a rotina de execução determinará a medida correta, solicitando a realização de replanejamento ou reparo do plano.

Segundo Haas, o exercício de um comando está associado a criação de um evento que representa o instante ou intervalo de tempo (podemos ter uma ação contínua) no qual ocorre uma troca de situações. A exemplo da teoria de simulação, a ocorrência sucessiva de eventos divide o tempo em pedaços irregulares denominados intervalos.

7.2 - Modelos Lineares de Planejamento

Para desenvolvermos planos que sejam capazes de determinar a execução apropriada de ações alcançando, por exemplo, objetivos conflitantes e observando as retrições de ordem temporal entre ações, é natural que sejam utilizadas programas de computadores com uma filosofia totalmente diferente dos programas convencionais (procedurais) e, até mesmo, dos sistemas especialistas tradicionais, em relação a representação e controle do conhecimento, haja visto a diferença marcante entre programas e planos segundo E. Charniak e D. McDermott [1985] .

Desta forma, vamos apresentar nesta seção alguns sistemas de planejamento lineares que foram précursores e estabeleceram os principais fundamentos para a construção de uma teoria de planejamento. Os sistemas serão apresentados de forma sucinta dando destaque a forma de representação do conhecimento e a realização do controle de ações.

7.2.1 STRIPS (R.E. Fikes e N.J. Nilsson [1971])

- Forma de Representação:

O sistema STRIPS utiliza para a representação do conhecimento uma linguagem de cálculo de predicados , de modo a configurar sucessivas trocas de estados em um determinado domínio de planejamento. Segundo N.J. Nilsson[1980] podemos considerar o sistema STRIPS como um sistema de produção no qual a base de dados (na forma de fórmulas atômicas) seja a combinação do estado atual associado a uma pilha de objetivos ou fórmula de predicados . Desta forma, as operações na base de dados, ou a execução de regras de transformação (operadores) se processam até o esvaziamento desta pilha.

No modelo STRIPS as regras de transformação são caracterizadas por uma fórmula pré-condicional que consiste de dois componentes : uma conjunção de literais com variáveis precedidas de um quantificador existêncial, e, uma lista de literais denominada lista de deleção. Desta forma, quando uma regra é aplicada, segundo uma descrição particular do estado ou descrição particular da base de dados que unifica com a fórmula pré-condicional , as instâncias de literais obtidas são deletadas da base de dados de acordo com a lista de deleção.

Finalmente, compondo o lado direito da regra, existe uma lista de adições ou acréscimo, constituída, também, de um conjunção de literais. Quando a regra é aplicada, as instâncias de literais produzidas são acrescidas ao estado ou base de dados pré-modificado pela lista de deleções, de acordo com a lista de acréscimos, derivando um novo estado na base de dados.

- Estratégia de Controle:

A estratégia básica de controle do sistema STRIPS consiste na execução das regras de transformação segundo um processo de encadeamento regressivo que procura satisfazer os objetivos existentes na pilha, um de cada vez , justificando a noção de planejamento linear, de uma forma oportunista (satisfazendo a descrição da base de dados) e seletiva (ordenando os componentes da fórmula não satisfeitos).

É possível implementar esta estratégia de controle na forma de uma busca em um grafo de estados , envolvendo, até mesmo, uma forma de função heurística que permita avaliar a escolha de operadores segundo o alcance dos objetivos. Entretanto, a implementação mais usual, proposta por N.J. Nilsson [1980], consiste de um procedimento recursivo com seleção não determinística e pontos de "backtracking" :

```
Procedimento  STRIPS ( G: objetivo );
Início
  Repita
    (* seleção não-determinística *)
    g <--  uma componente de G ( objetivo ) que não se liga
           a S ( estado );
    (* ponto de backtracking *)
    f <--  uma regra de transformação cuja lista de acréscimo
           contenha um literal que satisfaça  g;
    p <--  uma fórmula de pré-condições de uma instância
           apropriada de f;
    (* chamada recursiva *)
    STRIPS (p);
    S <- estado resultante da aplicação de f;
  Até S( estado ) se ligar a G ( objetivo );
Fim STRIPS;
```

- Contribuições:

Segundo T. Dean [1988] o sistema STRIPS apresenta como maiores desvantagens a não formulação de planos não-lineares e, também, a pouca expressividade da linguagem de planejamento utilizada. Entretanto, a caracterização de ação como o mapeamento, por parte de uma função de atualização , de um conjunto de sentenças, mundo ou estado em outro conjunto de sentenças, mundo ou estado foi uma contribuição de fundamental importância para a teoria de planejamento.

7.2.2 - GPS "General Problem Solver" (A.Newell e H.Simon [1963])

- Forma de Representação e Estratégia de Controle:

O sistema de solução de problemas GPS, anterior ao sistema STRIPS, apresenta uma técnica de planejamento bem semelhante (historicamente, o projeto do sistema STRIPS foi motivado pelo sistema GPS). Dada uma descrição de estado, S, e um objetivo, G, o processo de

escolha de uma regra de transformação visa reduzir a diferença entre S e G. Uma vez determinada a regra que mais reduz a diferença entre S e G, o sistema GPS trabalha recursivamente na pré-condição da mesma, derivando o seguinte procedimento:

```
Procedimento  GPS ( G: objetivo );
Início
  Repita
    (* ponto de backtracking *);
    d <-- a diferença entre S e G;
    (* ponto de backtracking *)
    f <--  uma regra de transformação que mais reduz  a
           diferença d;
    p <--  uma fórmula de pré-condições de uma instância
           apropriada de f;
    (* chamada recursiva *)
    GPS (p);
    S <- estado resultante da aplicação de f;
  Até S( estado ) ser igual a G ( objetivo );
Fim GPS;
```

- "Means-Ends Analysis":

O processo de identificação de diferenças e seleção da regra apropriada é conhecido como "means-ends analysis". Segundo N.J. Nilsson [1980], podemos considerar o sistema STRIPS como um caso particular do sistema GPS, permitindo que as diferenças entre G e S sejam aqueles componentes do objetivo não satisfeitos por determinada descrição do estado, e que aquelas regras cujas listas de acréscimo contenham literais que satisfaçam estas componentes, sejam consideradas relevantes para reduzir estas diferenças.

7.2.3 - Sistema HACKER (G.J. Sussman [1975])

- Técnica de Planejamento:

Dentro da linha de sistemas de planejamento lineares (ou seja, sistemas que usam um método de planejamento no qual o plano elaborado é resultado da concatenação de uma sequência de sub-planos, contendo, cada um, uma sequência de operações que possibilitam a satisfação de cada componente da pilha de objetivos), podemos destacar, também, o trabalho

desenvolvido por Sussman, que consiste de uma programa de computador denominado HACKER, voltado para a solução de problemas, particularmente, problemas cujo domínio seja referente ao mundo dos blocos.

Segundo T. Dean [1988], a exemplo da arquitetura SOAR, ver seção 6.1.6, o sistema HACKER é um exemplo de planejamento baseado em casos que requer alguns aspectos básicos referentes a teoria de aprendizado e a organização de memória .

Neste sentido, a organização de memória do sistema HACKER envolve uma biblioteca de planos, ou de respostas generalizadas. Desta forma, inicialmente, se um problema com uma conotação semelhante, já tenha sido, previamente, resolvido pelo sistema, é de se esperar que a nova solução seja, similarmente, parecida com a solução anterior. Entretanto, caso a antiga solução seja dependente de uma situação mais específica, que não se adapte a nova situação criada pelo novo problema, surge uma forma de impasse (semelhança com SOAR) que pode ser resolvida através de uma técnica que extenda a situação anterior, preservando, desta forma, as características principais da solução proposta anteriormente. Sussman considera esta técnica como a utilização de um plano de alto nível ou de um plano esqueleto que norteia o desenvolvimento de uma nova solução.

Como último recurso, considera-se a construção de uma proposta de programa, a partir de uma base de conhecimento e de uma biblioteca de técnicas de programação, que conduza a solução do problema. Para tanto, o programa proposto é, sucessivamente, criticado até conduzir a solução correta. Uma das motivações desta técnica é que, durante o processo de crítica, tanto as ocorrências de sucesso quanto as ocorrências de falha são generalizadas, para posterior utilização, aumentando o conhecimento do sistema de uma forma modular e iterativa.

- Contribuições:

Segundo T. Dean [1988], duas idéias importantes surgiram com o trabalho de Sussman:

. a possibilidade de se planejar através do uso e criticismo de planos já conhecidos, e

. o armazenamento da experiência em relação a construção de

planos, considerando os pontos positivos (caminhos até solução) e os pontos negativos (pontos de falha ou retorno).

Neste sentido, devemos considerar que, muitas vezes, só aprendemos certas soluções, ou abandonamos certos impasses, depois de repetidos erros ou de repetidas experiências negativas (problema do labirinto). Desta forma, achamos totalmente viável que o aprendizado derivado de experiências corretas, assim como o aprendizado derivado de experiências mal sucedidas, seja utilizado na elaboração de novas soluções.

- Aspectos Negativos:

Segundo Sussman, os principais aspectos negativos do sistema HACKER estão na limitação de conhecimento em relação a técnicas de solução de problemas e formas de representação e raciocínio (ao contrário de SOAR) favoráveis a solução de certos tipos de problemas. A dificuldade em formalizar o conhecimento intensivo e dependente do domínio para outros tipos de aplicações (que não seja do mundo dos blocos) e, ao mesmo tempo, se manter como uma arquitetura de caráter independente, é outro problema do sistema HACKER. Finalmente, Sussman destaca a falta de capacidade do sistema em "avistar a frente" no sentido de projetar planos, estratégicamente, eficientes e robustos.

A razão do sistema HACKER trabalhar com planos lineares se deve a sua extensibilidade ou capacidade de incorporar novos fatos ou mudanças. Por sua vez, a característica de extensibilidade, desejável aos sistemas inteligentes, requer, principalmente, a modularidade do conhecimento. Entretanto, para o conhecimento ser modular e extensível é necessário que a interação entre diferentes fontes de conhecimento seja restrita localmente, exigindo, desta forma, que a solução do problema não envolva a interação de passos ou a iteração entre diferentes fontes de conhecimento.

7.3 - Modelos Não-Lineares de Planejamento

7.3.1 - Planejamento Hierárquico

- ABSTRIPS (E.D. Sacerdoti [1974]):

O melhor trabalho que reflete o uso de planejamento hierárquico é o desenvolvimento do sistema ABSTRIPS, por Sacerdoti, comentado, anteriormente, na seção 4.3.2. Este sistema realiza o planejamento através da definição de uma hierarquia de espaços de abstrações, ignorando, iterativamente, literais de níveis mais baixos de abstrações das pré-condições contidas nas regras de transformação.

Deste modo, inicialmente, o sistema resolve o problema de planejamento, somente para literais com valores de criticalidade mais altos, estabelecendo um plano abstrato ou um plano esqueleto.

Em seguida, através de um processo de refinamento com introdução de detalhes, o sistema incorpora literais com níveis de criticalidade mais baixos, repetindo o processo, sucessivamente, até que todas as pré-condições estejam completas e o plano final tenha sido elaborado.

- Hierarquia e Reflexão em Sistemas de Planejamento:

Entretanto, conforme observações feitas por P.R. Young e P.E. Lehner [1986], a visão hierárquica de planejamento pode ser bastante ineficiente quando considerada em problemas reais de comando e controle, e, também, em problemas de planejamento dinâmico com "scheduling". Nestes problemas, ao contrário dos problemas com domínio estático, os níveis de detalhe do sistema podem comprometer a viabilidade e consistência de ações hierarquicamente mais elevadas, tornando sem utilidade o planejamento anteriormente realizado.

Contudo, vale a pena ressaltar que, quando falamos a respeito de sistemas de planejamento hierárquicos e, em particular, do modelo ABSTRIPS, não estamos referindo a todas as formas de controle hierárquico de sistemas. Segundo T. Linder [1988], é importante sabermos distinguir os diferentes tipos de hierarquia dentro de um sistema de planejamento. Por exemplo, a decomposição de objetivos em sub-objetivos e o uso de abstrações, como realizado no sistema SIGT, ver seção 6.2 , é uma forma de estabelecimento de níveis de hierarquia.

Similarmente, a proposta de uma estrutura hierárquica para o

SIGT, feita na seção 2.2.2,destacando os níveis de decisões estratégicas ou meta-planejamento, decisões táticas ou planejamento e , finalmente, de decisões operacionais ou execução, reflete , claramente, o emprego de um sistema de controle hierárquico através do uso do princípio da reflexão.

Para Linden,a reflexão constitui o princípio que distingue entre a execução da ação e o pensamento sobre como agir. Ou seja, se considerarmos a realização do planejamento, o processo de elaboração ou planejamento vem a ser uma reflexão sobre a execução, e o processo de meta-planejamento vem a ser uma reflexão sobre o planejamento. Neste sentido, a ocorrência de planejamento e execução de planos envolve o planejamento em diferentes níveis de abstração. Seguindo este ponto de vista, a tomada de decisões ou o emprego de ações faz parte do último nível de uma hierarquia de controle e gerenciamento.

Finalmente, queremos reafirmar o ponto de vista de Linden, segundo o qual, o planejamento se constitui em uma importante técnica para o desenvolvimento de sistemas que tenham flexibilidade e inteligência na tomada de decisões. Entretanto, o planejamento não é a única forma de alcançarmos sistemas flexíveis e inteligentes (como veremos adiante existe uma "trade-off" entre estes fatores), não sendo, também, um pré-requisito obrigatório a tomada de decisões (considere, neste caso, a ausência de reflexão).

7.3.2 - Planejamento Nao Linear

- Falha no Planejamento Linear:

A realização do planejamento linear, através da realização de uma sequência completa de operações, de modo a satisfazer , consecutivamente, cada componente da pilha de objetivos, nem sempre alcança o objetivo do problema.

Considere, como exemplo o problema no mundo dos blocos de alcançar o objetivo SOBRE(A,B) E SOBRE(B,C) a partir do estado inicial, descrito na figura 57.

estado inicial: objetivo:

figura 57: exemplo do mundo dos blocos.

Se optarmos, inicialmente , pela satisfação do componente SOBRE(A,B) , retirando o bloco C e colocando o bloco A sobre o bloco B, não sera possível colocarmos o bloco B, que se encontra sob o bloco A , sobre o bloco C, de modo a satisfazer o componente SOBRE(B,C).

De outro modo, se optarmos pela satisfação do componente SOBRE(Bb,C) , não teremos como satisfazer o componente SOBRE(A,B) pois o bloco A se encontrará sob o bloco C.

- Desenvolvimento de Planos Não-Lineares:

Um plano não linear, é definido como um plano composto de uma sequência de sub-planos ordenados parcialmente. Uma forma trivial de conduzirmos planos não lineares é não considerarmos a estratégia desenvolvida nos modelos lineares para a satisfação total dos componentes da pilha de objetivos, segundo o processo normal de expansão e seleção (do tipo LIFO "last in first out").

Desta forma, no problema anterior, se optarmos pela satisfação do componente SOBRE(A,B), retirando o bloco C , e, em seguida, optarmos pelo objetivo SOBRE(B,C), depositando o bloco B sobre o bloco C, poderemos, retornando ao componente SOBRE(A,B), satisfazer o objetivo do problema , colocando, finalmente, o bloco A sobre o bloco B.

Segundo N. Nilsson [1980], a estratégia correta seria construírmos uma árvore de busca, desenvolvida em função da permutação de uma sequência de operadores associados aos objetivos do problema que formariam um conjunto sem ordenamento. Desta forma, o processo de construção seria um processo regressivo, partindo da situação de

objetivo até atingir a situação ou estado inicial do problema. Este processo deve considerar, a cada vêz, a expansão de um único componente do conjunto de objetivos (que geraria novos componentes para o objetivo devido a existência de uma lista de pré-condições), através do uso do operador apropriado, seguindo um esquema de permutações que preserve os demais componentes.

Portanto, aquele conjunto de componentes ou de pré-condições , caracterizado como um vértice da árvore , que satisfizesse o estado inicial do problema , conduziria , de forma reversa , até a raiz (objetivo) da árvore o caminho solução , onde cada passo estaria associado ao respectivo operador selecionado.

A incoveniência deste método, esta relacionada as mesmas incoveniências dos processos que realizam buscas em largura, ou seja, o excessivo trabalho de geração de todos os caminhos de uma árvore, sendo que, a maioria , em geral, não conduz a satisfação do objetivo do problema.

- Estratégia "least-commitment" - NOAH (E.D. Sacerdoti [1977]):

. forma de representação:

Outra forma de realizarmos o planejamento não linear sem considerarmos a realização de todas as permutações consiste em aplicarmos a estratégia denominada " least-commitment", ou seja, definirmos a ordem na qual os operadores serão aplicados escolhendo, sempre, aqueles considerados mais convincentes segundo a situação ou estado específico.

O sistema NOAH , desenvolvido por Sacerdoti , realiza esta estratégia de planejamento estabelecendo um ordenamento parcial entre os operadores selecionáveis. O sistema, também, trabalha de forma hierárquica , através do uso de um contínuo processo de refinamento.

Para representar um plano, em andamento, NOAH utiliza uma rede procedural (E.D. Sacerdoti [1975]). Desta forma, a realização do planejamento fica caracterizado pelo desenvolvimento de sucessivas redes procedurais em diferentes níveis de abstração ou diferentes hierarquias, através de um processo de decomposição ou geração de sub-objetivos (

componentes) pela aplicação de operadores segundo a estratégia " least-commitment ".

. técnica de planejamento:

Para compreendermos melhor, esta estratégia de planejamento , sugerimos o seguinte exemplo ou problema:

Determinar um plano ou sequência de ações que conduza ao seguinte objetivo : OC (A,3) E OC (B,1). Considerando o seguinte estado inicial : OC (A,1) , OC (B,3) , DESOC (2) , DESOC (4), e DESOC (5), caracterizado pela figura 58.

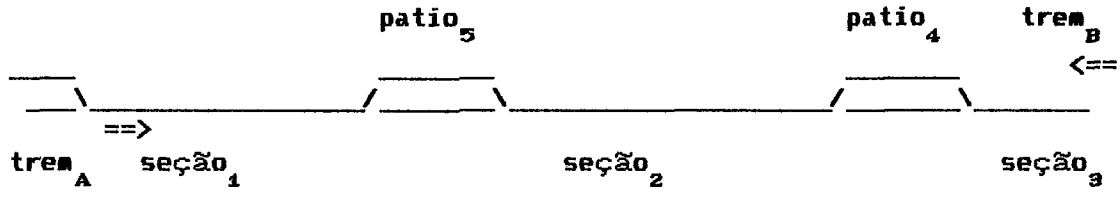


figura 58: situação de conflito.

O predicado OC (x,y) representa a ocupação da posição "y" pelo objeto "x" e tem como operador o passo AV (x,y) que avança o objeto "x" em direção a posição "y". Lógicamente, AV (x,y) tem como pré-condições os predicados DESOC(y) e PROX(x,y), ou seja, é necessário que a posição "y" esteja vazia e que o objeto "x" esteja adjacente a posição "y".

Inicialmente a rede procedural possui um único nó que determina o objetivo:

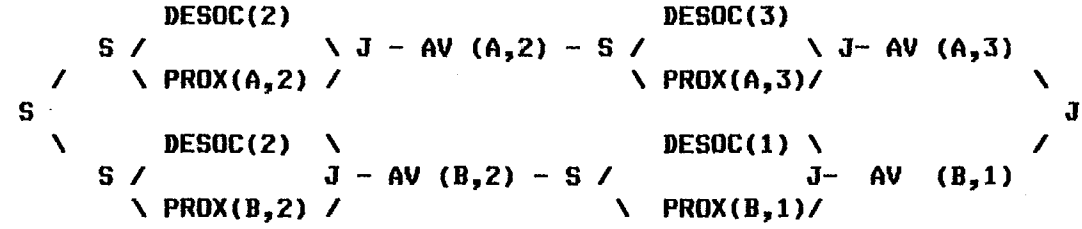
OC (A,3) E OC (B,1) nível 1

Em seguida, podemos forçar a decomposição do objetivo, gerando os dois componentes em paralelo:

OC (A,3)
/ \
S J nível 2
\ OC (B,1)/

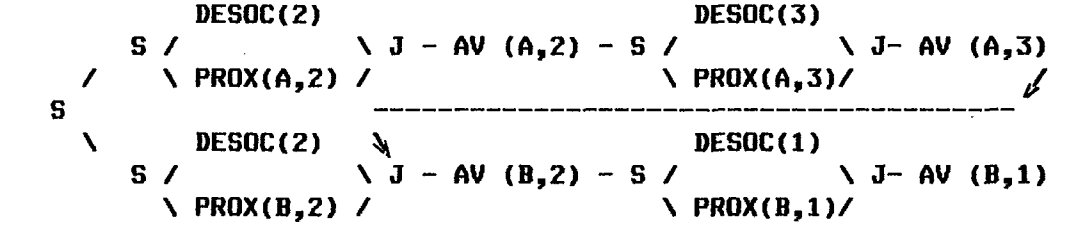
onde: nó S -> divisão, e
nó J -> junção.

Agora, o sistema considera os operadores, ou passos, que levam a satisfação dos objetivos, seguido de suas listas de pre-condições, determinando o nível 3:

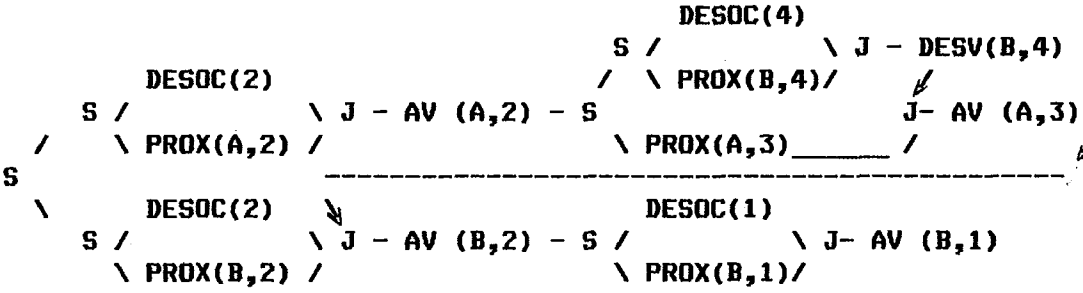


Neste ponto, através da realização de críticas, o sistema detecta uma interação entre os sub-planos existentes, pela ocorrência de várias menções de literais no mesmo plano. Neste caso, é necessário estabelecer um ordenamento entre as operações para resolver os conflitos potenciais existentes.

Seguindo nosso exemplo, verificamos a existência de duas possibilidades de ordenamento. Escolhendo satisfazer, inicialmente, o primeiro componente , OC (A,3), teremos, a rede procedural no nível 4:



Entretanto, na análise das pré-condições, verificamos a não satisfação da condição DESOC(3), já que neste estado é verdadeiro o predicado OC(B,3) . Desta forma, através de um processo de refinamento e uso do passo DESV (B,4), que desvia o trem B para a posição 4, obtemos a rede procedural no nível 5:



Finalmente, após observamos a existência de certos predicados que são verdadeiros para a situação inicial do problema , obteremos o seguinte plano final :

AV (A,2) - DESV (B,4) - AV (A,3) - AV (B,2) - AV (B,1).

Podemos considerar, que o plano de ações acima é parcialmente ordenado segundo os componentes do objetivo do problema e, portanto, não linear (não confundir não linearidade de planos com paralelismo de ações). Entretanto , se optassémos, no nível 3, pela outra estratégia de ordenamento, teríamos como plano final :

AV (B,2) - DESV (A,5) - AV (B,1) - AV (A,2) - AV (A,1).

Os problemas a respeito da existência de ações paralelas em planos e, também, a respeito da dominância entre alternativas, serão discutidos com profundidade em seções posteriores deste capítulo.

Neste exemplo, podemos observar que o sistema NOAH trabalha com uma estratégia muito semelhante a descrita por N.J. Nilsson [1980], através de um esquema de regressão com geração de nós e refinamentos sucessivos. Entretanto, através da geração paralela de operadores a cada nível de abstração e através do uso do procedimento de crítica (verificação de conflitos, pré-condições redundantes, etc.) no processo de seleção , é possível estabelecer um ordenamento ideal de operadores , resultando em uma técnica de planejamento mais eficiente.

Portanto, vimos que foi possível a realização de um planejamento não linear sem a criação de uma árvore de busca. Entretanto, dada a impossibilidade de realização de "backtracking" , a técnica de planejamento empregada pelo sistema NOAH nem sempre se mantem eficiente. A.Tate [1977] propos um sistema semelhante ao sistema NOAH , denominado NONLIN, mas que permite a realização de "backtracking" ao ponto onde é realizado o ordenamento de operadores, em situações de impasse, resolvendo problemas considerados insolúveis pelo sistema NOAH.

. Considerações:

Uma crítica pessoal que podemos fazer ao sistema NOAH é que o

mesmo não trata, explícitamente, o problema de recursos, restrições em variáveis e, também, não inclui capacidade para raciocínio temporal (embora o ordenamento de nós na rede procedural estabeleça as informações necessárias para o ordenamento de ações) e raciocínio sob incerteza gerada, principalmente, pela imprecisão e ausência de informações.

Queremos ressaltar, também, a incapacidade do sistema em trabalhar com várias versões de planos em função de diferentes estratégias de ordenamento que sejam, lógicamente, viáveis. Desta forma , não é possível estabelecer o melhor plano segundo algum critério de avaliação.

Estas questões, que afetam a quase totalidade dos sistemas de planejamento, serão, também, discutidas com a devida profundidade, em seções posteriores deste capítulo.

- Estratégia "Constraint-Posting" - MOLGEN (M. Stefik [1981a]):

. técnica de planejamento:

Segundo M. Stefik, este processo de planejamento consiste na adição sucessiva de restrições, de forma a refletir a existência de interação entre sub-problemas que compõe abstrações de espaços durante o processo de planejamento hierárquico. Neste sentido, as restrições são formuladas dinâmicamente e propagadas durante o planejamento, de modo a coordenarem a solução de subproblemas que se tornam quase independentes.

Portanto, quando da elaboração do plano final, dado a utilização de um processo de satisfação de restrições, temos um plano considerado viável, que atende a todas as restrições, descartando, desta forma a necessidade do uso de "backtracking", tal como utilizado no sistema NONLIN. Esta técnica de planejamento , aumenta, também, a eficiência do processo de planejamento hierárquico, devido a capacidade de trabalhar com interação de planos.

O desenvolvimento deste sistema de planejamento está, íntimamente, associado ao desenvolvimento de um programa de computador, denominado MOLGEN (M. Stefik [1981b]) , desenvolvido na Universidade de Stanford e voltado para a realização de experimentos científicos no

campo de genética molecular.

. introdução de restrições no sistema SIGT:

Se considerarmos o desenvolvimento de planos no sistema SIGT como um processo de planejamento de um sistema formado por problemas quase independentes que representassem a movimentação dos trens na malha ferroviária, teríamos a existência de fortes interações , proporcionadas pelas situações de conflito que se formariam ao longo da realização do planejamento.

Desta forma, podemos considerar que a nossa técnica de planejamento, como dito na seção 3.6.4, se fundamenta nesta estratégia, pois a profundidade dos planos de horários em relação ao período de planejamento, cresce, na medida em que, dinâmicamente, introduzimos restrições que permitem tratarmos as interações resultantes das situações de conflito.

A exemplo do sistema MOLGEN, consideramos em nosso planejamento as três operações básicas que envolvem o processo de introdução de restrições:

. formulação de restrições: caracterizada no SIGT pela utilização das restrições de precedência da abstração matemática do problema;

. propagação de restrições: caracterizada no SIGT pelas funções de atualização das tabelas de tempos, e

. satisfação de restrições: caracterizada no SIGT pela instanciação ou geração dinâmica de uma tabela de valores numéricos que atendem , parcialmente, as restrições do problema.

Em nosso sistema, tivemos uma grande facilidade de implementação desta estratégia de planejamento, ao definirmos uma abstração matemática , a qual nos permitiu lidar com valores numéricos , matrizes, função de variáveis, etc. Entretanto, podemos considerar, também, o uso do sistema de produção, ao longo do processo de planejamento, que determina preferências na escolha de operadores, como uma forma de introdução de restrições as variáveis simbólicas do sistema , consideradas em nosso problema como instâncias de "frames", possuindo um conjunto de atributos e predicados.

7.3.3 - Planejamento e Otimização

- Dominância entre Planos:

Segundo M.P. Wellman [1988], o processo de planejamento incremental proposto por M. Stefik consiste em acrescentar restrições a classes de planos candidatos, parcialmente completos, ou seja, planos que satisfazem apenas parcialmente o conjunto de restrições .

No planejamento tradicional o problema básico consiste em identificar um plano viável ou satisfatório que conduza a determinado objetivo. Entretanto, podemos, de uma forma mais complexa, considerar o problema de planejamento como a identificação do melhor plano que, obviamente, conduza ao objetivo estabelecido.

Porem, as técnicas tradicionais de planejamento, descritas anteriormente, não se preocupam com este aspecto do planejamento relativo a otimização de planos. Isto se deve, principalmente, a não existência de uma estrutura que estabeleça uma forma de organização de planos parciais durante a realização do planejamento, bem como, de um esquema de classificação que estabeleça uma forma de dominância ou de preferência entre planos candidatos.

Desta forma, Wellman propõe uma forma de organização ou especializações de planos envolvendo a formação de classes, sub-classes a partir do ordenamento de ações primitivas, ou seja:

Se $A = \{ a_1, \ldots, a_n \}$ se refere a um conjunto de ações primitivas, podemos definir $\Sigma = A^*$ como uma linguagem de planos. Desta forma,é possível representarmos um espaço de planos na forma de um grafo , ver figura 59, onde cada sub-classe é representada por um conjunto de ações sequenciais (totalmente ou parcialmente ordenadas) .

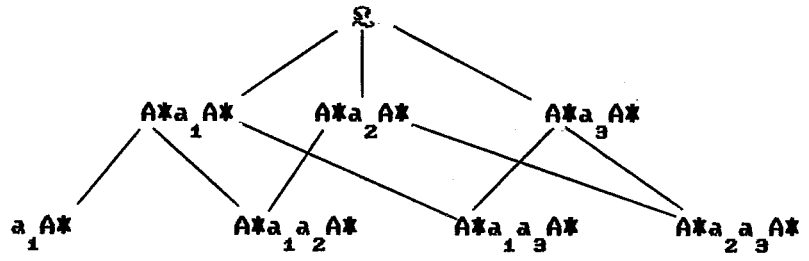


figura 59: grafo de planos parciais.

Observando, que:

. o vértice $A^*a_1A^*$ se refere ao conjunto de planos com pelo menos uma ação a_1;

. o vértice a_1A^* se refere ao conjunto de planos ou subclasse comecando com a ação a_1;

. o vértice $A^*a_1a_2A^*$ se refere ao conjunto de planos ou subclasse onde a ação a_1 é seguida pela ação a_2.

Wellman considera a técnica de planejamento por introdução de restrições de M. Stefik como capaz de supotar a realização de uma busca no espaço de representação de planos. Ou seja, a adição de várias restrições permite grande flexibilidade no esquema de avaliação e classificação de planos parciais. Ao contrário, a estratégia de escolha do operador mais convincente, ou a adição, somente, de restrições bem justificadas, como no sistema NOAH, só possibilita a avaliação de planos completos.

Dentro deste raciocínio, Wellman propõe um procedimento de busca ou de construção de planos que preserve todo o universo de planos parciais candidatos (parcialmente viáveis), evitando a realização de "backtracking" e possibilitando a escolha de melhores planos. Neste sentido, podemos encarar a realização de planejamento não como a construção de um simples plano, mas sim, como a exploração de um espaço de planos parciais candidatos, orientada em função das propriedades de admissibilidade e das provas de dominância. Para tanto, torna-se necessário definirmos:

1 - Uma linguagem de planos ;

2 - Uma linguagem de restrições para a representação de planos parciais;

3 - Uma linguagem de domínio de forma a representarmos o efeito das ações ("frame problem") e as relações de preferência, e

4 - Uma prova de dominância.

Como ilustração, apresentaremos alguns exemplos de Wellman , relacionando esta teoria de planejamento a problemas de otimização:

Por exemplo, se considerarmos o espaco R^ncomo uma linguagem de planos e o domínio do problema como a região viável de pontos determinados por um conjunto convexo de restrições. Se todos os pontos

viáveis fossem comparados a possíveis planos completos, as relações de preferência seriam determinadas através da avaliação dos pontos pela função objetiva, e as condições de otimalidade seriam a prova de dominância que possibilitaria encontrarmos o melhor plano , relativo ao ponto de ótimo do problema. Ao utilizarmos um algoritmo, que produz uma sequência de pontos que conduz a solução ótima do problema, estamos realizando uma busca no espaço de pontos (planos), orientada, principalmente, pela viabilidade do sistema de restrições (preservação do conjunto ativo de variáveis ou restrições) e , também, segundo o critério de otimização (análise dos multiplicadores).

Entretanto, o problema geral de otimização em R^n não faz uso da técnica de planejamento incremental, ou do desenvolvimento de planos parciais. Este processo de planejamento fica melhor caracterizado em problemas de programação discreta, como exemplo: programação inteira, onde a utilização de algorítmos do tipo "branch and bound" requer, claramente, a introdução de restrições formando planos parciais, fazendo uso de uma prova de dominância explícita, através da atualização de limites e da realização de um processo de busca exaustivo.

Dando continuidade a seu trabalho, Wellman faz uma análise do sistema de planejamento TWEAK (D. Chapman [1987]) quanto a sua capacidade face a esta nova interpretação de planejamento.

- Otimização de Planos no Sistema SIGT :

A forma de planejamento incremental efetuada pelo sistema SIGT é totalmente compatível com as considerações feitas por M.P. Wellman em relação a exploração de um espaço de planos candidatos, através de refinamentos sucessivos, pela adição de restrições, verificando as propriedades de admissibilidade e incluindo relações de preferência de modo a caracterizar o aspecto de dominância.

A linguagem de planejamento utilizada para construção de um espaço de planos , bem como, a linguagem de restrições incluindo a formulação , propagação e satisfação , e a caracterização do domínio, foram bem formalizados através do uso de um sistema físico simbólico associado a um modelo matemático que inclui uma representação de estados, operadores, processos de seleção , funções de avaliação, etc.

, já amplamente discutidos em capítulos anteriores de nosso trabalho.

Ademais, a adoção de um modelo de planejamento e "scheduling" nos permitiu desenvolver uma estrutura de controle durante o processo de planejamento, que possibilitou a determinação do melhor plano. Na seção 3.4.2, discutimos a prova de dominância, quanto a satisfação de um conjunto de objetivos na forma conjuntiva e um conjunto de restrições estruturais do problema de "scheduling" segundo o ponto de vista da programação multi-objetiva "fuzzy".

Na seção 3.6.3, extendemos este raciocínio, desenvolvendo uma função de avaliação única, admissível, que permitiu, através do uso do algoritmo A*, o estabelecimento de preferência entre planos candidatos e a determinação do melhor plano.

Finalmente, na seção 6.1.4, a exemplo da arquitetura SOAR, estabelecemos uma extensão aos critérios de determinação de preferências na escolha de operadores, de modo a satisfazer novas restrições não consideradas, anteriormente, e possibilitar o uso de heurísticas.

Na proxima seção, vamos discutir o sistema de planejamento TWEAK, destacando a sua principal contribuição a teoria de planejamento que foi o desenvolvimento de um "criterio da verdade" que delineou o nosso processo de planejamento. Este critério, colocado em nosso sistema sob a forma de algumas propriedades clássicas do planejamento, serviu como base teórica para a técnica de planejamento incremental, quanto ao estabelecimento da ordem na qual as ações foram inseridas (ver seção 3.6.4) , e, também, quanto a viabilidade da representação da tabela de tempos.

7.3.4 - Critério da Verdade - TWEAK (D. Chapman [1987])

O sistema de planejamento TWEAK é um elaborador de planos não lineares que tambem adota a técnica de introdução de restrições, sendo voltado para a solução de problemas que apresentam um conjunto de objetivos na forma conjuntiva. Este sistema, apesar de ser um sistema de planejamento recente , não incorpora um tratamento mais adequado as restrições temporais e a representação do tempo, a exemplo do sistema

DEVISER (S.A. Vere [1983]) e da teoria de tempo e ação de J.F. Allen [1983],[1984] e A.Haas [1985].

Devemos considerar, também, que todos os sistemas de planejamento discutidos até então, são de linha estratégica (T. Dean [1988]), não dando ênfase as questões de planejamento com incerteza. Nesta linha de planejamento estratégico, os sistemas assumem, preliminarmente, que o agente responsável pelo planejamento tenha conhecimento completo da situação na qual o plano foi elaborado, bem como , do efeito futuro do emprego das ações. Estas considerações, para problemas reais, parecem pouco eficientes. Entretanto, a discussão de questões envolvendo a realização de planejamento temporal, bem como, sob incerteza, serão consideradas, somente, em seções posteriores deste capítulo.

- Representação de Planos:

No sistema TWEAK, um plano, também, é considerado como uma sequência de passos, ou ações, parcialmente ordenados, aplicados a objetos. O efeito das ações é completamente descrito, a exemplo do sistema STRIPS, a partir do uso de conjuntos ou listas de pré e pós-condições que antecedem ou sucedem o emprego de ações. A relação de preferência ou indiferença entre planos é assumida na forma mais simples , segundo M.P. Wellman [1988] Ou seja, se um plano alcança determinado objetivo e o outro plano não, então o primeiro plano é preferível quanto ao segundo. Entretanto, se os dois planos alcançam ou não alcançam o objetivo, então são ditos indiferentes entre si.

Basicamente, o sistema TWEAK trabalha em um problema, tendo sempre em mão, um plano incompleto . O que significa uma especificação parcial de um plano completo. Neste caso, existem duas maneiras de um plano estar incompleto:

1 - A ordem entre os passos ou ações pode estar incompletamente especificada;

2 - Os passos ou ações podem estar incompletamente especificados.

Desta forma, dado que um plano total existe, o sistema tenta tornar completo um plano parcial e incompleto, através da introdução de

restrições caracterizadas por um conjunto de operadores específicos.

No primeiro caso, o sistema especifica uma mudança de ordenamento entre os passos do plano. Para tanto, são utilizados operadores na forma de restições temporais, que estabelecem um reordenamento entre os passos, ou até mesmo, sugerem a introdução de novos passos.

No segundo caso, temos que estabelecer uma relação de equivalência entre variáveis e constantes . Ou seja, em um plano completo, cada variável que aparece em uma lista de pré e pós-condições deve retratar-se ou estar codesignada a uma constante específica.

- Caracterização do Espaço:

O sistema TWEAK representa possíveis estados como situações caracterizadas por um conjunto de predicados. Um plano tem uma situação inicial, que descreve o domínio no qual se insere o planejamento no momento inicial, e uma situação final, que descreve o estado que será compatível com a realização do planejamento e alcance dos objetivos.

Associado a cada passo de um plano, temos uma situação de entrada, representada por um conjunto de predicados que condicionam a execução da ação, e uma situação de saída que deve ser compatível com o efeito da ação executada. Em um plano completo, a situação de entrada de cada passo é compatível com a situação de saída do passo anterior , sendo a situação de saída do último passo a situação final do planejamento.

- Alcance de Objetivos:

O elaborador de planos TWEAK é definido como um procedimento não determinístico que procura alcançar os objetivos aplicando, sucessivamente, uma sequência de operadores que visam completar planos parciais. Desta forma, o "loop" de controle escolhe, repetidamente, um objetivo não satisfeito, e tenta alcançá-lo através da execução de um passo. Para tanto, o sistema interpreta o "critério da verdade" de forma não determinística , escolhendo operadores que provocam modificações no plano no sentido de tornar mais restrito os planos

parciais de modo a alcançar todos os objetivos na forma completa.

O critério da verdade é descrito no trabalho de Chapman, na seguinte forma:

" Uma proposição "p" é, necessáriamente, verdadeira em uma situação "s", se e somente se, duas condições forem satisfeitas:

1 - Existe uma situação "t", igual ou necessáriamente, anterior a "s", na qual "p" é considerada verdadeira;

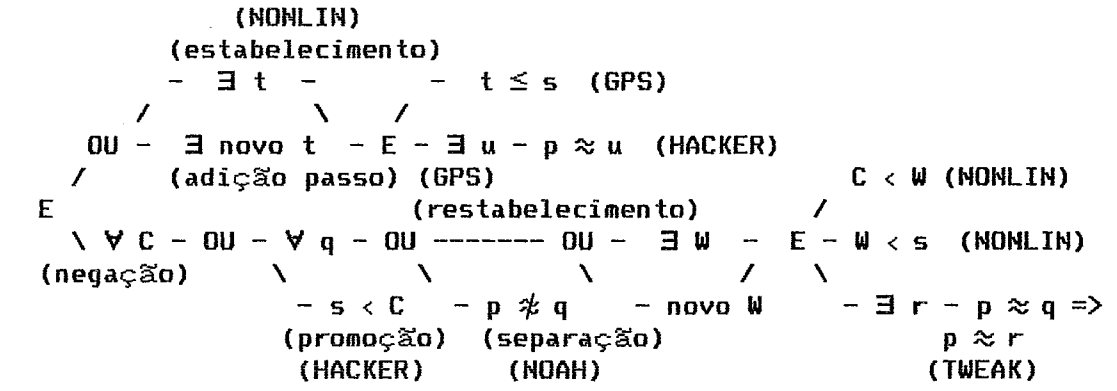
2 - Para cada passo C, possivelmente anterior a "s", e para cada proposição "q", que possivelmente codesigna "p", que é negada em C, deve existir um passo W, necessáriamente entre C e "s", que estabelece "r", uma proposição tal que: se "q" e "p" se codesignam, então "p" e "r" também se codesignam ".

Desta forma, podemos considerar o passo caracterizado na situação "t", como o passo estabelecedor da proposição "p". O passo C como o passo de negação da proposição "p", e, finalmente, o passo W como o passo restabelecedor da proposição "p".

Comparado aos teoremas de H.A.Kautz e E.P.D. Pednault [1985], ver seção 3.6.4, o critério da verdade fornece uma versão mais completa, no sentido de possibilitar o restabelecimento de uma proposição. D. Chapman [1987] considera esta operação de modificação de planos como a única novidade do sistema TWEAK, considerando que as outras operações de modificação, decorrentes do critério da verdade, já tenham sido utilizadas em sistemas de planejamento anteriores.

Na figura 60, (D. Chapman [1987]) é apresentado o procedimento não determinístico , ao qual nos referimos anteriormente, associado ao critério da verdade, representado através de uma árvore geradora.

A respeito da propriedade de codesignação, podemos dizer que uma proposição e verdadeira em uma situação, se ela codesigna com uma proposição verdadeira que e membro da situação. Podemos afirmar, tambem , que uma proposição e falsa em uma situação, se ela codesigna com uma proposição falsa que e membro da situação.



Simbologia:

∃ existe um ;
E conjunção;
OU disjunção;
≤ temporal precedência ou igualdade;
≈ codesigna;
u proposições necessáriamente verdadeiras em "t";
∀ cada um;
≉ não codesigna;
=> implica;
q pós-condições de C;
r pós-condições de W;

figura 60: procedimento de escolha não determinístico.

Quanto ao procedimento não-determinístico, podemos considerar, ainda, um caso particular de modificação de planos, não representado, denominado demoção, e que consiste em escolher "t" como uma situação de saída do passo W, tornando, neste caso, sem efeito, o passo C.

- Estrutura de Controle :

O processo de elaboração de um plano no sistema TWEAK é considerado como a realização de uma busca através de um espaço de caminhos alternativos produzidos pelo processo anterior de alcançe de objetivos (interpretação iterativa do critério da verdade desencadeando na escolha não determinística de operadores que modificam planos).

Entretanto, a escolha de certo objetivo a ser alcançado, bem como a escolha de uma operação de modificação de planos são tarefas que exigem algum processo de ordenamento. Alem disto, a possível introdução de passos ou restrições, pode gerar um sistema de restrições inconsistente, levando a necessidade da realização de "backtracking".

No sistema TWEAK, e empregado uma estrutura de controle caracterizada pela realização de uma busca em largura, associada a realização de "bactracking" com dependencia direta (R.M. Stallman e G. J. Sussman [1977]) ao inves do "bactracking" tradicional de dependencia cronologica.

Segundo M.P. Wellman [1988], o poder de planejamento do sistema TWEAK reside nas propriedades de dominância que são aplicadas, as quais permitem restringir o espaço de planos a um pequeno conjunto de possíveis completações de planos parciais. Desta forma, é possível ao sistema reconhecer que certas classes de planos não precisam ser exploradas pelo fato de serem dominadas, mesmo considerando que os mesmos possam produzir planos válidos.

Finalmente, queremos destacar que o sistema TWEAK, pelo fato de realizar uma busca exaustiva no espaço de planos, segundo as operações de modificação, termina sempre com uma solução viável, que , realmente, resolve o problema. Caso o sistema não termine ou termine com uma falha , é porque a solução do problema, verdadeiramente, não existe.

- Considerações:

No sistema SIGT, a utilização desta prova de dominância explícita a partir do uso do critério da verdade de D. Chapman é, matemáticamente, interpretada na seção 3.3.3 quando determinamos um processo de fixação de variáveis do tipo 0-1 condicionado a existência de conflitos no procedimento de solução do problema de sequenciamento de trens. Convém lembrarmos, que este processo de fixação de variáveis, reduz, enormemente, a complexidade inicial do problema , ou, análogamente, o espaço de planos viáveis.

Desta forma, foi possível o desenvolvimento de uma técnica de planejamento incremental orientada pela solução de conflitos , que conduz ao alcance de objetivos. Consideramos, também, a interpretação do critério da verdade, pelo fato de haver uma forte interação entre sub-problemas ao longo do processo de planejamento, haja visto a ocorrência sucessiva de conflitos

Para resolvermos estes conflitos, tornou-se necessário a introdução de um passo ou ação que estabelece o cumprimento das

restrições de precedência do problema.

- Representação de Ações e Sinergia:

No sistema TWEAK, bem como nos outros sistemas de planejamento, o aspecto mais negativo em relação a representação de ações, se deve ao fato de não se permitir que o efeito das ações dependa da situação na qual o plano é realizado ou executado. Para tanto, seria necessário uma representação de estados com pré e pós-condições que fossem funcionalmente dependentes de situações particulares. Entretanto, mesmo considerando a existência desta representação, veremos que o sistema TWEAK falha na execução do plano.

No trabalho de D. Chapman [1987] é apresentado um problema de blocos onde o critério da verdade falha. O problema consiste na realização do seguinte plano, ver figura 61.

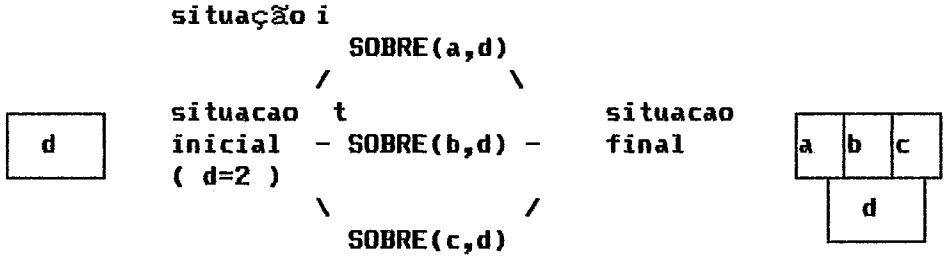


figura 61: exemplo do mundo dos blocos.

Se considerarmos como pré-condição do passo SOBRE(x,y) o fato de y possuir espaço vazio pelo menos igual a 1, e considerarmos, também, que o efeito da ação reduz o espaço vazio em uma unidade, podemos afirmar que o plano acima falhou, levando a uma situação final indesejável.

Entretanto, vamos considerar a realização dos passos na situação "i". Neste caso, concordando com o critério da verdade, deve haver uma situação "t", necessáriamente anterior a "i", na qual a pré-condição é verdadeira. Não deverá haver, também, em uma situação entre "t" e "i" a ocorrência de um passo que nege a pré-condição.

Isto é perfeitamente correto, se considerarmos a situação "t"

como a situação inicial e observarmos que cada possível passo situado entre "t" e "i" reduz o espaço do bloco "d" em apenas uma unidade, não negando, portanto, a pré-condição.

Este tipo de situação é denominado por Chapman de sinergia, ou seja, somente o efeito acumulado de passos ou ações é que afeta lógicamente a pré-condição da ação desejável que conduz ao objetivo. Observe que este problema não acontece em planos lineares devido a existência de um ordenamento total entre ações na consecução de objetivos.

Uma forma de resolvermos o problema de sinergia seria considerarmos a utilização de um quantificador de recursos, ou variáveis de estado, que atualizassem, dependentemente da situação, a informação de quantidade associada. Esta solução é proposta em nosso sistema, na solução dos problemas de capacidade e utilização de recursos, caracterizados pela ocupação de pátios de cruzamento e linhas singelas , ver seção 5.5.3 .

A bem da verdade, certa potencialidade do sistema SIGT, foi possível devido ao fato de nossa arquitetura ser voltada para uma classe específica de problemas, ou seja, podemos considerar o SIGT como um sistema ou arquitetura dependente de um certo domínio de planejamento, que consiste de problemas de gerenciamento e controle de sistemas dinâmicos com operações de planejamento e "scheduling".

Finalmente, queremos relembrar a existência do "frame problem" que é colocado por D. Chapman [1987] como determinar de forma eficiente uma implementação do critério da verdade. Em nosso sistema, esta implementação foi alcançada, em grande parte, devido a visão multi-disciplinar sob a qual tratamos o problema de gerenciamento e controle de trens.

7.3.5 - Planejamento com Recursos - SIPE (D.E. Wilkins [1984])

- Introdução:

SIPE é um sistema de planejamento, independente do domínio, que suporta a realização de planos hierárquicos não-lineares de uma forma

automática ou interativa. A diferença deste sistema em relação aos sistemas de planejamento anteriores, como exemplo NOAH, NONLIN e TWEAK, é a capacidade do mesmo em raciocinar sobre recursos, raciocinar sobre interações entre ações paralelas e, também, quanto a possibilidade do usuário observar e controlar o processo de geração de planos.

Contudo, a representação de planos e o processo de geração são semelhantes ao sistema NOAH, através do uso de redes procedurais , planejamento em níveis de hierarquia , uso de críticas para resolver interações entre sub-problemas e expansão de nós objetivos . O sistema SIPE permite, também, a introdução de restrições de forma a prover descrições parciais de objetos ou variáveis considerados no planejamento.

Apesar da forte expressividade deste sistema entre os sistemas de planejamento de linha estratégica, o mesmo também é desprovido de uma teoria que possibilite a realização, com eficiência, de planejamento com incerteza e planejamento temporal. Entretanto, a questão de execução do planejamento, envolvendo a realização de monitoração e replanejamento é tratada de forma adequada neste sistema e será abordada, posteriormente, em nosso trabalho.

- Representação de Planos:

O sistema SIPE, através de sua técnica de representação baseada em redes procedurais, incorpora as facilidades dos sistemas baseados em "frames", permitindo o continuísmo das propriedades e atributos de objetos que não sofram variações sob o efeito das ações ao longo do planejamento (não considerando a utilização de listas de deleção e acréscimo como no sistema STRIPS), e , por outro lado, mantendo a eficiência do formalismo do cálculo de predicados na representação dos operadores (pré-condições, efeitos, etc.) que estabelecem as ações, e dos atributos e relacionamentos de objetos que possam variar sob o efeito das ações.

Desta forma, representando as propriedades invariantes do domínio, separadamente, o sistema SIPE aumenta a eficiência do processo de dedução. Entretanto, não é permitido , durante a realização do planejamento, a destruição e criação de objetos.

Convem lembrarmos, que a utilização conjunta de estruturas de representação de estados mais eficientes, e estruturas de representação da parte invariante do domínio, foi considerada em nosso sistema pela integração do sistema gerador de planos ao sistema baseado em conhecimento contituido de "frames" e regras de produção, conforme considerações já feitas ao longo do trabalho.

A rede procedural é constituída de diversos nós que estabelecem uma hierarquia, a exemplo da representação em redes semânticas , permitindo a realização das propriedades de instanciação e herança. Entretanto, em alguns nós da rede procedural, que representam objetivos, operadores e pré-condições , não são permitidas estas propriedades.

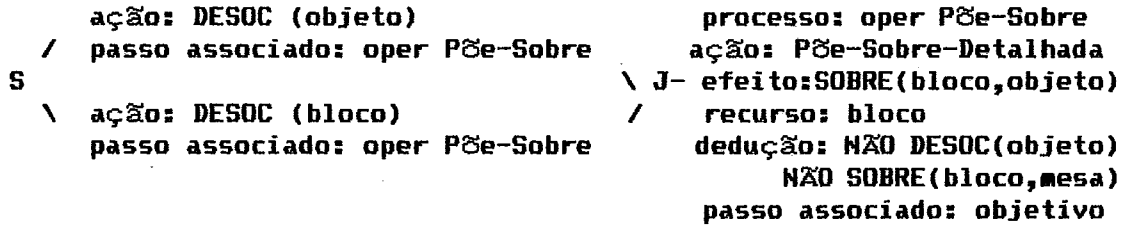
No sistema SIPE, os operadores contém, essencialmente, ver figura 62, informações sobre objetos que participam das ações como argumentos ou recursos, do objetivo associado, do efeito provocado pela ação e da lista de pré-condições existentes. Possuem, também, no caso de operadores dedutivos, axiomas que permitem a dedução de proposições a respeito do domínio , considerando a representação de efeitos colaterais.

```
operador: Põe-Sobre;
argumentos: bloco, objeto que não seja bloco;
objetivo: SOBRE (bloco,objeto);
nó: (* instrução para expanção do nó na rede procedural *)
   paralelo
      caminho 1:
         objetivo: DESOC (objeto);
         argumentos: objeto
      caminho 2:
         objetivo: DESOC (bloco);
         argumento: bloco
   fim;

   processo (*  ação associada ao objetivo *)
   ação: Põe-Sobre-Detalhada;
   recurso: bloco;
   efeitos: SOBRE(bloco,objeto);
   dedução: NÃO DESOC(objeto)
            NÃO SOBRE(bloco,mesa)
fim.
```

figura 62: descrição do operador Põe-Sobre.

Como podemos verificar, normalmente, não existem pré-condições especificadas na descrição de operadores, na verdade, estas pré-condições aparecem como objetivos (caminho 1 e caminho 2) que vão preceder o operador na rede procedural caso seja necessária a expansão do nó, ou seja:



Também, podemos observar que a existência da ação Põe-Sobre-Detalhada , estabelece uma hierarquia de abstrações , requerendo a realização de planejamento em um nível de detalhamento inferior.

- Uso de Restrições:

No sistema SIPE, operadores possuem listas de recursos e argumentos que devem se ligar aos recursos e argumentos, nos nós que são gerados pela expansão. A exemplo do sistema MOLGEN, o sistema SIPE tem capacidade de construir descrições parciais de objetos não especificados, através do uso de uma linguagem de restrições. Ou seja, as variáveis do planejamento, ainda não instanciadas, podem ser parcialmente descritas através da adição de restrições, retardando o processo de instanciação e prevenindo de escolhas incorretas.

Dentre os tipos de restrições que podem ser acrescidas, podemos destacar restrições nos valores de atributos de objetos e variáveis (instâncias de objetos) , no relacionamento entre classes e objetos, objetos e variáveis, e nos valores de predicados que tenham variáveis como argumentos.

Por exemplo, através do uso de restrições que impõe valores em predicados que possuem a variável como argumento , é possível determinar a priori, considerando o conhecimento de predicados verdadeiros , um número restrito de escolhas para a instanciação da variável. Da mesma

forma, na verificação da consecução de objetivos, observando as restrições associadas as variáveis, e na realização de críticas na determinação de interações entre ações paralela, através da análise de compatibilidade entre as restrições.

Segundo Wilkins, a principal vantagem no uso de restrições, reside no fato de que não é necessário definir, explícitamente, cada propriedade de um objeto como um predicado associado as pré-condições de operadores na elaboração do plano.

Podemos considerar que, no sistema SIGT, através do uso de variáveis de "frames", não necessitamos especificar , detalhadamente, os atributos das variáveis , nas pré-condições dos operadores utilizados ao longo do planejamento. Na verdade, grande parte do esforço na tarefa de redução da complexidade de busca realizada no planejamento, é conseguida através da determinação de preferências em operadores a partir de inferências realizadas na base de conhecimento, garantindo, de forma eficiente, a viabilidade do processo de instanciação e manipulação de variáveis ao longo do planejamento.

- Raciocínio sobre Recursos:

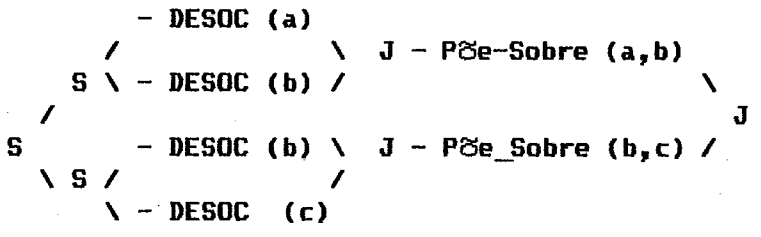
No sistema SIPE, a representação de um operador inclui a possibilidade de se associar variáveis envolvidas na ação , como recursos do domínio, como exemplo: bloco , do operador da figura 62.

Desta forma, o sistema pode, automáticamente, aferir a disponibilidade do recurso, como pré-condição a realização da ação. Portanto, no sistema SIPE, a viabilidade de aplicação de operadores quanto a existência de recursos, é analisada previamente, permitindo a redução do espaço de busca pela escolha de operadores que não produzem conflitos.

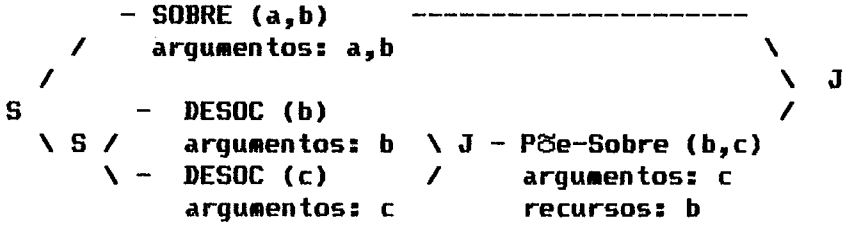
Seja, por exemplo, os dois planos representados a seguir, que consideram a colocação ,de um bloco "b" sobre um bloco "c" e, em seguida, a colocação de outro bloco "a" sobre o bloco "b".

Para que o plano sem recursos seja viável, é necessário que seja estabelecido um ordenamento nas ações de modo que o bloco "b" seja colocado sobre o bloco "c" antes que o bloco "a" seja colocado sobre o

bloco "b". Sistemas como NOAH e NONLIN fazem este ordenamento através da realização de críticas, avaliando o efeito de múltiplas ações paralelas nos valores das respectivas pré-condições.



plano sem recursos.



plano com recursos.

No sistema SIPE, quando algum objeto é declarado como recurso (plano com recurso), o sistema não permite que o mesmo seja considerado, também, como recurso, em outra ação paralela que tenha o mesmo como argumento.

Portanto, no problema acima, o sistema executará, primeiramente, a ação Põe-Sobre (b,c), liberando a utilização do recurso, e retardando a expansão do objetivo SOBRE(a,b). Contudo , na verificação de outras formas de interações, que não as provocadas pela utilização simultânea de recursos, o sistema SIPE faz uso do mesmo processo de críticas utilizado nos sistemas NOAH e NONLIN.

A exemplo do sistema SIPE, permitimos em nosso sistema, como já dissemos, a quantificação de recursos em variáveis globais associadas a "slots" de "frames" , e a verificação a priori, através de inferências na base de conhecimento, no módulo de regras situado no nível de restrições do sistema , ver seções 5.4.4 e 5.5.3 .

- Paralelismo entre Ações:

A existência de ações paralelas em um plano deve, acima de tudo,

observar a existência de interações no que se refere a consecução de objetivos.

Desta forma, Wilkins define como interações ruins aquelas interações que afetam, de forma conflitante, a consecução dos objetivos. E, como interações amigáveis, aquelas que favorecem a consecução de objetivos, ou seja, o alcance do objetivo de uma ação, é favorecido pelo exercício da outra ação e vice-versa, como exemplo: as ações paralelas necessárias a consecução dos objetivos do exemplo anterior, associadas ao operador Põe-Sobre.

Contudo, caso as interações entre ações paralelas não sejam amigáveis, é necessário fazermos uma ordenação, como forma de restabelecer a validade do plano.

No sistema SIGT, podemos, por exemplo, considerar como ações paralelas, o desvio simultâneo do $trem_B$ para o $pátio_4$, e o avanço do $trem_A$ na $seção_2$, no plano relativo a solução do problema descrito na figura 58, na seção 7.3.2 .

Um estudo mais detalhado sobre a existência de ações paralelas em nosso sistema, bem como, a sua interpretação segundo o critério da verdade de Chapman , será visto na seção seguinte.

- Estrutura de Controle:

O sistema SIPE realiza uma busca em profundidade até uma dada profundidade associada a expanção de nós objetivos. Segundo Wilkins, embora o sistema apresente uma estrutura de representação e manipulação de planos eficiente, o conhecimento de controle, como exemplo a realização de meta-planejamento, é limitado em função do aspecto interativo.

Ou seja, através da realização de um planejamento interativo, o usuário pode dirigir e direcionar o processo de elaboração de planos, através do planejamento de operações. Para tanto, o sistema apresenta uma interface gráfica e amigável que faz uso de janelas, menus de opções, visualização gráfica, etc.

Similarmente, em nosso sistema, consideramos a realização de

planejamento interativo , através do uso do modelo de simulação, discutido na seção 6.4.4, que produz uma busca em profundidade orientada segundo o usuário e as inferências realizadas na base de conhecimento. Desta forma, é possível adotarmos o planejamento de operações , através da escolha de ações apropriadas, cujos efeitos são visualizados gáaficamente. Neste sentido, Wilkins destaca a capacidade do sistema SIPE em explorar o uso de alternativas de ações em paralelo.

Para tanto, Wilkins apresenta um mecanismo de contexto que permite o estabelecimento de restrições em variáveis em partes específicas do plano, associadas a nós de escolha. Desta forma, é permitido ao usuário retroceder e avançar no planejamento, quanto a escolha de alternativas , restaurando ou recuperando, sempre, as restrições associadas as variáveis.

Estas operações , também presentes no sistema SIGT, pela conservação de uma lista de estados, não podem ser feitas em sistemas que realizam "backtracking", devido ao fato de que, as descrições construídas durante a expansão de um nó , são removidas durante o processo de "backtracking" antes que nova alternativa seja investigada.

7.3.6 - Paralelismo entre Ações no Sistema SIGT:

O sistema SIGT visto como um modelo de planejamento, tem a capacidade de elaborar planos constituídos por um conjunto de ações parcialmente ordenadas que formam planos não lineares associados as situações de conflito mais um conjunto de ações consequentes, que formam planos paralelos associados as ações de sequenciamento, apesar da existência de um único agente no planejamento.

- Construção de Planos segundo o Critério da Verdade:

Para compreendermos melhor o esquema de elaboração de planos no sistema SIGT, e a existência de paralelismo entre ações, vamos considerar um problema de planejamento semelhante ao problema da seção 7.3.2, representado na figura 63, segundo a filosofia do sistema TWEAK.

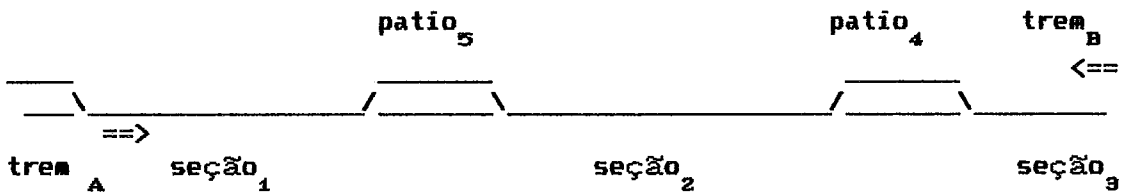


figura 63: situação de conflito.

objetivo: OC(A,3) E OC(B,1);

estado inicial: OC(A,1), OC(B,3), DESOC(2), DESOC (4), DESOC(5), PROX(A,2), PROX(A,4), PROX(B,2) e PROX(B,4), dado pela figura 63.

plano inicial:

objetivo OC (A,3)

DESOC(2), PROX(A,2) | DESOC(3), PROX(A,3)

[AV(A,2)] —S→ [AV(A,3)]

OC(A,2), PROX(A,3), DESOC(1) | OC(A,3) DESOC(2)

objetivo OC (B,1)

DESOC(2), PROX(B,2) | DESOC(1), PROX(B,1)

[AV(B,2)] —S'→ [AV(B,1)]

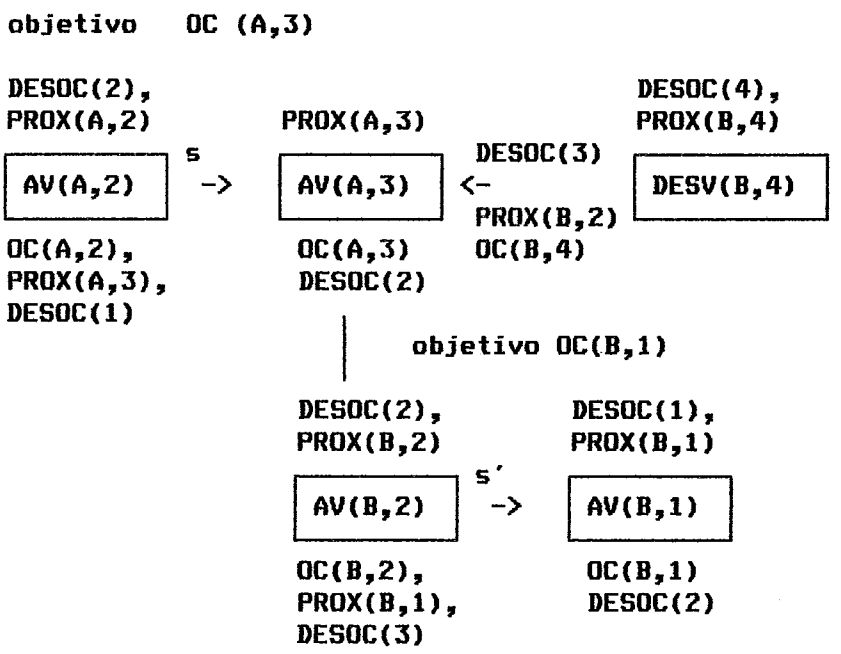
OC(B,2), PROX(B,1), DESOC(3) | OC(B,1) DESOC(2)

Inicialmente, após a interpretação do critério da verdade, surge a necessidade de evitar a interação entre a consecução dos objetivos OC(A,3) e OC(B,1). Para tanto, será necessário estabelecer uma restrição temporal, ou ordenamento de passos de forma a evitar a ocorrência de conflito. Isto deve ser feito através da "promoção" da situação "s'" relativa ao passo AV(B,2).

Desta forma, o sistema tentará alcançar, inicialmente, o objetivo OC(A,3) através do uso dos passos AV(A,2) e AV(A,3) por simples estabelecimento, já que podemos considerar como "t" a situação inicial.

Em seguida, após a nova interpretação do critério da verdade é verificada a existência de uma anomalia, a proposição DESOC(3) é falsa, obrigando a introdução de um novo passo, DESV(B,4), como forma de estabelecimento, gerando a seguinte situação:

plano modificado:



Finalmente, por demoção devido a introdução do passo , podemos alcançar o objetivo OC(B,1) através da realização dos passos AV(B,2) e AV(B,1).A demoção se justifica pelo fato de que a situação "t" referente ao estabelecimento do passo AV(B,2) foi possível graças ao predicado PROX(B,2) resultante da situação de saída do passo DESV(B,4).

Portanto, através do uso da promoção e da adição de um passo,

foi possível completar o plano inicial e alcançar os objetivos OC(A,3) e OC(B,1).

- Exemplo de Paralelismo com um Único Agente:

Agora, vamos considerar apenas o avanço de um trem na malha ferroviária, segundo a situação descrita na figura 64.

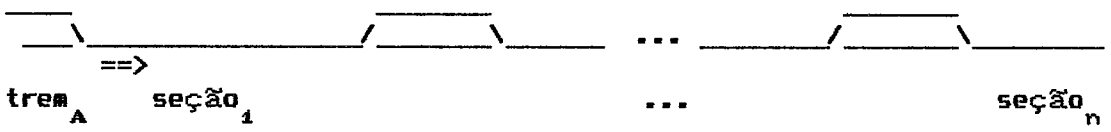


figura 64: situação de avanço.

Desta forma, teremos como plano inicial:

objetivo OC (A,n)

DESOC(2), PROX(A,2)		DESOC(n), PROX(A,n)
AV(A,2)	s -> ... ->	AV(A,n)
OC(A,2), PROX(A,3), DESOC(1)		OC(A,n) DESOC(n-1)

Ao analisarmos o critério da verdade, verificamos que só poderemos executar os passos de sequenciamento, ou o passo AV(A,2) na situação "s" se garantirmos a não existência de um passo C , possivelmente anterior a situação "s", que torne o predicado DESOC(2) negativo. O predicado PROX(A,2), bem como os predicados PROX(A,3) ,..., PROX(A,n) , serão verdadeiros por simples estabelecimento, sendo uma consequência natural das ações de sequenciamento.

Neste caso , o passo C , só existirá quando ocorrer uma situação de conflito , sendo o estabelecimento da pre-condição resolvido por promoção (ver exemplo anterior). Felizmente, como já dissemos, o sistema SIGT tem capacidade de reconhecer eficientemente as situações de conflito através do uso de uma tabela de tempos e, portanto, interpretar de forma racional o critério da verdade. Observe que , no caso da não existência do passo C ou da situação de conflito, o critério da verdade

será plenamente atendido, possibilitanto a execução contínua das ações de sequenciamento.

Para que o critério da verdade seja, constantemente, atendido durante o nosso processo de planejamento, é suficiente identificarmos e resolvermos as situações de conflito, de acordo com a ordem lógica de ocorrência; para tanto, como já dissemos anteriormente, devemos sempre identificar o primeiro conflito na ordem cronológica na tabela de tempos. Agindo desta forma, garantimos a não ocorrência de um passo C , de negação, durante a execução das ações de sequenciamento.

Neste ponto, queremos fazer algumas observações a respeito do paralelismo entre ações, em nosso sistema:

Suponha que em determinado instante do planejamento, existam varias situações associadas a execução de passos de sequenciamento. Se considerarmos , que estas situações pertencem ao intervalo definido pelos instantes de ocorrência de dois conflitos consecutivos, poderemos considerar a execução paralela destes passos, já que não existe neste intervalo aberto de tempo, a ocorrência de conflitos ou interação entre sub-problemas. Lógicamente, durante a elaboração de planos, a execução de ações paralelas é ditada, pela solução consecutiva de conflitos, a exemplo da realização de eventos paralelos em nosso esquema de simulação.

Na verdade, o processo de elaboração de planos no SIGT , determina, somente, a solução de conflitos. Este problema, ou seja, a solução de conflitos no sistema real, esta associado a existência de um único agente, denominado controlador, ver seções 2.3.3 e 6.4.4. Desta forma, as ações de sequenciamento que permitem aos trens avançarem, livremente, ao longo da linha, são consideradas como efeito do plano ou das ações de resolução de conflitos, sendo, portanto, desvinculadas do único agente do sistema.

Se observarmos a estrutura hierárquica do SIGT na seção 2.2.2 , veremos que estas ações estão situadas no nível de decisões operacionais, não requerendo, portanto, reflexão, ou um nível abstrato por parte do sistema de planejamento. Desta forma, disassociando estas ações do agente de planejamento, permitimos a existência de paralelismo durante o processo de elaboração de planos.

A bem da verdade, durante a fase de execução do planejamento, o controlador estabelece, previamente, as condições de avanço dos trens pelo posicionamento de chaves, resolvendo, consecutivamente, os conflitos e permitindo a existência de um real paralelismo que permite o avanço simultâneo de trens. Na situação extrema, de não ocorrência de conflitos, não teríamos necessidade de planejamento, já que os trens circulariam livremente, sem a intervenção do controlador.

- Desenvolvimento de Planos Esqueletos:

Segundo P.R. Young e P.E. Lehner [1986], um agente com uma certa experiência na tarefa de desenvolvimento de planos, para determinada classe de problemas utiliza, sempre, inicialmente, um procedimento generalizado , que já tendo sido utilizado em problemas similares, facilita o desenvolvimento de um plano esqueleto, que consiste, lógicamente, de uma sequência de passos representada em um nível mais elevado na hierarquia de abstrações.

Como exemplo, na tarefa de elaboração de planos de horários para o problema de sequenciamento de trens, podemos citar o procedimento utilizado pelo planejador ou analista, que utiliza, inicialmente, uma grade ou diagrama espaco × tempo onde estão representadas apenas as restrições de sequenciamento do sistema. Desta forma, através de um processo de refinamento, que consiste na solução sucessiva de conflitos, o analista elabora a grade final.

Se considerarmos este ponto de vista como correto, ou seja, colocando as ações associadas as retrições de sequenciamento como pertencentes a um nivel mais alto na hierarquia de planejamento, estamos contradizendo o raciocínio exposto anteriormente. Entretanto, ao que nos parece, o processo correto de elaboração de planos, não necessita, a priori, da estruturação de um plano esqueleto baseado nas restrições de sequenciamento.

Se recordarmos à seção 3.6.3, veremos que a manutenção de uma grade com o percurso total dos trens, se, por um lado, facilita a tarefa do analista responsável pela elaboração do plano, para nós, permite, somente, avaliar a componente heurística da função de avaliação, garantindo a utilização de algoritmos de busca que propiciam a elaboração de planos ótimos segundo a formulação estabelecida para o

problema. Portanto, reafirmamos, novamente, o ponto de vista anterior, o qual caracteriza as ações de sequenciamento como dependentes do plano de solução de conflitos.

7.3.7 - Organização de Sistemas de Planejamento

Segundo E.A.Rich [1983], para resolvermos de forma eficiente muitos problemas em Inteligência Artificial, é necessário utilizarmos uma estrutura de organização de programas que permita a existência de cooperação entre diferentes fontes de conhecimento e o compartilhamento de múltiplas bases de dados. É universalmente aceito, que as formas mais eficientes de lidar (no sentido de reduzir) com a complexidade de sistemas é através da decomposição modular e do emprego de hierarquia de abstrações.

Neste sentido, vamos descrever duas das principais formas de organização de sistemas de planejamento e solução de problemas em Inteligência Artificial : arquiteturas de agendas e quadro-negros, que fazem uso dos processos de decomposição e abstração; e destacar a organização do sistema de programação SIGT quanto a influência destas arquiteturas.

- Agendas - MOLGEN (M.Stefik [1981b]):

Uma agenda, segundo M. Stefik, consiste de um mecanismo de controle do conhecimento, sendo uma extensão natural da arquitetura resumida de computadores convencionais ou de sistemas de produção. Basicamente, uma agenda contém um interpretador que, repetidamente, seleciona e executa tarefas que estão organizadas em uma agenda. A memória, que guarda dados, cálculos temporários e resultados, é ,frequentemente, estruturada como uma rede semântica. A execução de tarefas causa, lógicamente, uma mudança no estado da memória através da introdução de novos valores e alteração ou deleção de valores antigos.

Entretanto, a forma de selecionar tarefas, não obedece, a exemplo dos mecanismos de controle de execução de instruções em programas convencionais, o sequenciamento e emprego de desvios. Entre as mais variadas sugestões para a seleção de tarefas, destaca-se o uso

de valores numéricos ou prioridades associados a tarefas. Porém, como estes valores podem variar ao longo do processo de interpretação,de modo a refletir as condições de mudança , propõe-se, também, o emprego de funções de estimativa de prioridades, no lugar de simples valores numéricos.

O aspecto mais negativo desta técnica se deve ao número elevado de interações entre possíveis argumentos das funções associadas as tarefas. Ou seja, para computarmos os valores das funções, teríamos que lever em conta todas as interações exixtentes entre grupos de tarefas , o que teria, certamente, crescimento exponencial em função do tipo de combinação e do número de tarefas existentes.

Certamente, a proposta mais eficiente para o processo de interpretação de agendas, se deve a M. Stefik [1981b], que define uma hierarquia de agendas, considerando a existência de meta-níveis entre os componentes de um problema. Desta forma, uma agenda teria como interpretador uma nova agenda que constituiria um nível mais alto do problema onde seria representado o controle necessário a resolução do problema original. Neste sentido, as meta-tarefas em um problema de nível mais alto são voltadas para a seleção e execução de tarefas no problema original. Como vemos, esta estrutura de organização poderia ser aplicada, recursivamente, gerando, como consequência, uma hierarquia de agendas aninhadas segundo vários níveis de abstração, ver figura 65.

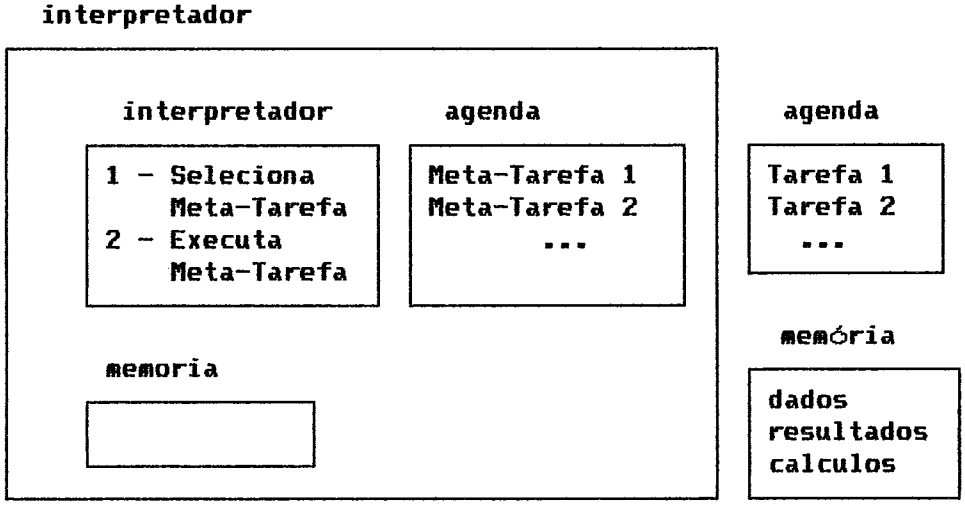


figura 65: estrutura de agendas aninhadas.

M. Stefik considera a utilização desta estrutura de controle hierárquica no sistema de planejamento MOLGEN. Neste sistema, Stefik define três níveis de espaço:

1 - Nível mais inferior: este nível compreende o espaço de domínio do problema que representa o conhecimento básico a respeito de objetos e operações necessários a realização de passos ou aplicações de operadores a nível de laboratório;

2 - Nível médio: este nível compreende o espaço de projeto que representa o conhecimento sobre o projeto de planos, onde são definidos os operadores responsáveis pela formação de planos abstratos e pela propagação de restrições;

3 - Nível mais alto: este nível compreende o espaço estratégico que contém o conhecimento ou passos necessários a criação e execução dos passos no espaço de projeto.

Neste sentido, M. Stefik considera que os passos em nível de projeto, planejam a execução dos passos definidos no nível de laboratório, e que os passos estratégicos, através da realização de um meta-planejamento, programam a execução dos passos de projeto.

Por fim, temos também, no sistema MOLGEN, a existência de um interpretador associado ao espaço estratégico, que forma a estrutura ou "loop" mais externo de controle do programa MOLGEN, que direciona a execução dos passos estratégicos.

Quanto ao conhecimento representado no espaço estratégico, podemos destacar o uso de heurísticas que são aplicadas no sentido de possibilitar a tomada de decisões onde o conhecimento é considerado imcompleto. Podemos destacar, também, o uso da estratégia "least-commitment" ou a aplicação do passo mais convincente que caracteriza a tomada de decisões de modo conservativo, ou seja, derivada da interação entre sub-problemas e da propagação de restrições.

- Quadro-Negros - HERSAY (L.D. Herman e outros [1980]):

A arquitetura de quadro-negros é constituída de um conjunto de módulos independentes ou fontes de conhecimento (FC), que contém o conhecimento específico do problema acrescido de um quadro-negro que forma uma estrutura de dados na qual as fontes de conhecimento tem

acesso. Esta arquitetura é proveniente do sistema de reconhecimento de vozes HERSAY II (L.D. Herman, F. Hayes-Roth, V.R. Lesser e D.R. Reddy [1980] , e possui um conjunto de hipóteses que tratam o processo de solução de problemas como um processo oportunista.

Segundo B. Hayes-Roth [1985], este processo considera que:

1 - Todos os elementos de solução, gerados ao longo do processo, são gravados em uma estrutura global de dados, comumente chamada de quadro-negro. Em geral, esta estrutura apresenta duas dimensões associadas , respectivamente, aos níveis de abstração, e ao tempo de execução ou a dimensão do tempo, que estabelece a existência de intervalos;

2 - Os elementos de solução são gravados no quadro-negro por processos independentes chamados fontes de conhecimento (FC). Também, em geral, estas fontes de conhecimento consistem de regras do tipo condição -> ação, situadas em diferentes níveis de abstração. Associado a cada regra existe um conjunto de procedimentos que observam certas condições que descrevem situações nas quais a fonte de conhecimento possa contribuir para o processo de solução do problema.

Desta forma, se as condições forem atendidas, a ativação da fonte de conhecimento determina a criação ou modificação dos elementos-solução no quadro-negro. Apesar das diferentes implementações deste tipo de arquitetura, cada ativação de uma fonte de conhecimento é condicionada a existência de um evento, e produz um único registro de ativação de fonte de conhecimento ou RAFC , similar a um ítem ou tarefa de uma agenda;

3 - A cada ciclo de solução do problema, um mecanismo de programação é responsável pela escolha de um único RAFC e executa a sua ação. A exemplo dos sistemas de produção , a ocorrência de um evento pode determinar um conjunto de fontes de conhecimento candidatas a ativação. Entretanto, o mecanismo de programação selecionará somente uma para ser executada. Obviamente, a execução deste RAFC proporcionará a produção de novos eventos, estabelecendo, desta forma , o ciclo de reconhecimento-seleção-ação. Entre os critérios de seleção mais comuns destacam-se: a confiabilidade da fonte de conhecimento, a credibilidade da informação e a importância da ativação da FC no processo de solução do problema.

A respeito da existência do mecanismo de programação, B. Hayes-Roth destaca em seu artigo que o problema de controle (que vem a ser a escolha apropriada da ação a ser executada, por um sistema inteligente, no processo de solução de problemas) deve ser tratado, pelos sistemas de Inteligência Artificial, como um problema de planejamento em tempo real. Desta forma, o sistema deve operacionalizar a realização de um possível controle inteligente, mantendo uma conduta objetiva, que atenda os seguintes pontos:

1 - Adotar decisões de controle explícitas;

2 - Decidir quais ações a executar, levando em consideração, ao mesmo tempo, as decisões que são favoráveis ou desejáveis e as decisões que são viáveis ou permissíveis;

3 - Adotar uma heurística de controle de tamanho variável, que ora se refira a classe de ações e que ora se refira a ações específicas;

4 - Adotar uma heurística de controle que focalize as ações favoráveis ao processo de solução do problema;

5 - Adotar, manter ou desfazer-se de heurísticas de controle segundo a dinâmica (troca de situações) do problema;

6 - Decidir como integrar diferentes heurísticas de controle de importâncias condicionadas a um conjunto de objetivos e ou situações;

7 - Elaborar, dinâmicamente, planos estratégicos, e

8 - Raciocinar sobre as relativas prioridades de ações de domínio e de ações de controle.

Sendo assim, B. Hayes-Roth[1985] propõe uma arquitetura de controle de quadro-negro (ACQN) , aplicada a um problema de planejamento diário de multi-tarefas, que extende a arquitetura básica de quadro-negro, observando as seguintes considerações:

1 - A ACQN representa, explícitamente, o problema de controle e o problema relativo ao domínio, o conhecimento e soluções. Ou seja, a arquitetura especifica as fontes de conhecimento necessárias a solução destes problemas , bem como os quadro-negros associados;

2 - A ACQN integra o problema de escolha de ações de controle e ações de domínio , em um simples "loop" de controle. Este "loop" básicamente compreende: a atualização do conjunto de RAFC, a escolha de

uma RAFC e a interpretação e execução de sua ação, gravando no quadro-negro suas consequências;

3 - A ACQN articula, interpreta e modifica a sua conduta e a representação de seu próprio conhecimento;

4 - A ACQN adapta o conhecimento utilizado na solução do problema, sua aplicação e o "loop" básico de controle, segundo a dinâmica do problema. Ou seja, a conduta geral do sistema é inteiramente determinada pelas interações sucessivas entre as fontes de conhecimento de domínio e controle, envolvendo, continuamente, a solução do problema de domínio e do problema de controle, e

5 - A ACQN realiza a tarefa de solução de problemas em um mecanismo uniforme , através de um processo de cooperação e interação entre fontes de conhecimento independentes.

- Modelo de Planejamento Oportunístico:

As técnicas de planejamento discutidas até então, como os sistemas de planejamento ABSTRIPS, NOAH e TWEAK, se baseiam na técnica de projeto denominada "top-down". Neste caso, o processo de planejamento é unidirecional e constituido, de duas fases, relacionadas, respectivamente, a projeção e refinamento de planos.

Segundo R. Wilensky [1983] a projeção consiste de um mecanismo através do qual detectamos problemas em um plano incompleto, ou seja, a projeção quantifica todas as instâncias de um determinado plano abstrato , de forma a saber qual a melhor maneira de se fazer o refinamento ou a transformação do plano.

No sistema SIGT, a realização de projeção (não confundir com o fase de elaboração de planos, citado na seção 2.1.3) envolve a determinação dos conflitos existentes na tabela de tempos, onde são guardados os eventos passados e futuros, ver seção 6.4.4, que delineam parcialmente as ações de planejamento.

Ao contrário dos sistemas anteriores, os autores apresentam um modelo de planejamento, de arquitetura quadro-negro, que possue uma conduta direcionada ao mesmo tempo pelo alcance dos objetivos e pela ocorrência de informações. Esta técnica de planejamento é caracterizada

como oportunista no sentido de que este tipo de planejamento implica em decisões que podem ocorrer em pontos não adjacentes do espaço de planejamento. Ou seja, diferentes decisões podem ser tomadas, em diferentes níveis de abstração e diferentes intervalos, interagindo sinérgicamente na tentativa de formação de planos.

Portanto, esta técnica de planejamento necessita de uma estrutura de representação de planos associada a arquitetura quadro-negro, já que os modelos de representação dos sistemas anteriores não permitem esta forma de planejamento considerada heterárquica.

Desta forma, considerando a atividade de planejamento como um processo multi-direcional, onde o refinamento de um determinado plano pode apresentar detalhes que implicariam na concepção de partes mais abstratas do projeto, o sistema deve apresentar uma conduta, durante o processo de geração de planos, caracterizada pelo uso combinado de técnicas de projeto "top-down" e "bottom-up".

M. Stefik [1981b] considera que o modelo de B. e F. Hayes-Roth descreve dois paradígmas fundamentais de planejamento: hierárquico e oportunístico. O paradígma hierárquico consiste, lógicamente, no processo de refinamento de planos que faz uso de abstrações. Enquanto que, o paradígma oportunístico, constitui um processo bidirecional e heterárquico que permite que sub-planos sejam desenvolvidos, independentemente, em diferentes níveis de abstração, para uma eventual cooperação sinergética na concepção do plano final. Segundo Stefik, a idéia de oportunismo é manifestada, análogamente, no processo de adição de restrições do sistema MOLGEN.

No sistema SIGT, vimos que, pela interpretação do critério da verdade, o processo de refinamento, determinado pela adição de restrições, deve atender, rigorosamente, a dependência lógica entre conflitos , que culmina com a pesquisa cronológica da tabela de tempos. Entretanto, as ações sub-sequentes, que determinam, por exemplo, o avanço sequencial dos trens, são produzidas em intervalos consecutivos de tempo , em diferentes pontos da linha férrea, contribuindo, sinergicamente, para a elaboração do plano final.

Podemos considerar, também, a verificação de condições ou situações específicas, relativas a restrições físicas e operacionais,

durante o processo de planejamento, como uma forma de realimentação constante ou "feedback" ao processo de planejamento, caracterizando, segundo Stefik, uma conduta "bottom-up" ou um planejamento do tipo bidirecional.

De fato, se considerassemos a verificação destas restrições, somente após estabelecermos um plano de solução de conflitos (abstrato) completo, poderíamos ter consequências desastrosas para o planejamento, implicando, até mesmo, na perda de quase todo o plano realizado.

Neste trabalho, procuramos descrever apenas alguns aspectos da arquitetura e de alguns sistemas do tipo quadro-negro associados a questão do planejamento oportunístico e a representação de planos. Atualmente, a partir do desenvolvimento do sistema Hersay-II, em meados da década de 70, tem se desenvolvido pelo menos 30 sistemas baseados nesta filosofia de organização, com aplicações variadas, incluíndo a área de sistemas militares e robótica, com especial ênfase em sistemas de arquiteturas distribuídas e paralelas.

O crescimento dos sistemas do tipo quadro-negro foi tão significante, que resultou na realização do primeiro encontro de trabalhos em sistemas quadro-negro em 1987, patrocinado pela Associação Americana de Inteligência Artificial e pelo Centro de Tecnologia Avançada da Boeing, tendo lugar em Seattle, EUA. R. Dodhiawala e outros[1989], apresentam um breve resumo deste evento, sugerindo, também, para um estudo mais detalhado a respeito da evolução de sistemas quadro-negros, a consulta ao trabalho de H.P. Nii [1987].

- Organização do Sistema SIGT:

Para o sistema SIGT, foi proposta uma certa modificação na estrutura de quadro-negros a exemplo do sistema desenvolvido por G.J.Stroebel, R.D. Baxter e M.J.Denney[1986]. Como podemos verificar, ver Apendice II, a estrutura do programa é dividida em conjuntos de módulos independentes, representando formas de abstração funcional ou formas de abstração de dados (tipo abstrato de dados) . Neste sentido, consideramos os módulos funcionais divididos em dois tipos:

. aqueles que implementam conhecimento na forma funcional, como

exemplo: módulos de regras, função de avaliação, etc., representando fontes de conhecimento do sistema, e

. aqueles que implementam estruturas de controle , como exemplo: algoritmos, interpretador de regras, etc.

Por outro lado, os módulos que implementam tipos abstratos como tabelas, estados, elementos da memória de trabalho, "frames", etc., são considerados como formadores de um quadro-negro geral, onde as estruturas de dados que compõem este quadro são partilhadas por todos os módulos do sistema. É interessante observarmos que a especificação da interface dos módulos, determina sua funcionalidade, bem como o grau de interação com outras partes do sistema, sendo, também, completamente isolada dos detalhes de implementação.

Completando a organização proposta para o SIGT no capítulo II , descrevemos o esquema representado na figura 66, onde destacamos a presença dos principais módulos do sistema acompanhado das principais funções.

Neste sentido, podemos observar que, hierarquicamente, o sistema SIGT é o sistema mais abrangente , compreendendo o sistema de planejamento, as funções de monitoração e replanejamento, que serão descritas neste capítulo, e um modelo de simulação.

Podemos observar também, que o sistema forma uma arquitetura inteligente. Esta arquitetura, envolve um sistema baseado em conhecimento e o sistema gerador de planos. O sistema baseado em conhecimento, por sua vêz, tem como componentes um sistema de produção e uma base de "frames" e "regras".

Finalmente, destacamos o sistema gerador de planos, que constitui um sistema físico simbólico, construído a partir de uma extensão da modelagem matemática "fuzzy" proposta para o problema de sequenciamento de trens.

7.4 - Planejamento com Incerteza

Conforme já dissemos, os modelos de planejamento vistos até então, como os sistemas STRIPS, NOAH, TWEAK, MOLGEN e outros, são considerados de linha estratégica. Estes modelos assumem, inicialmente,

total conhecimento da situação na qual os planos são elaborados, bem como do efeito das ações que compreendem o planejamento.

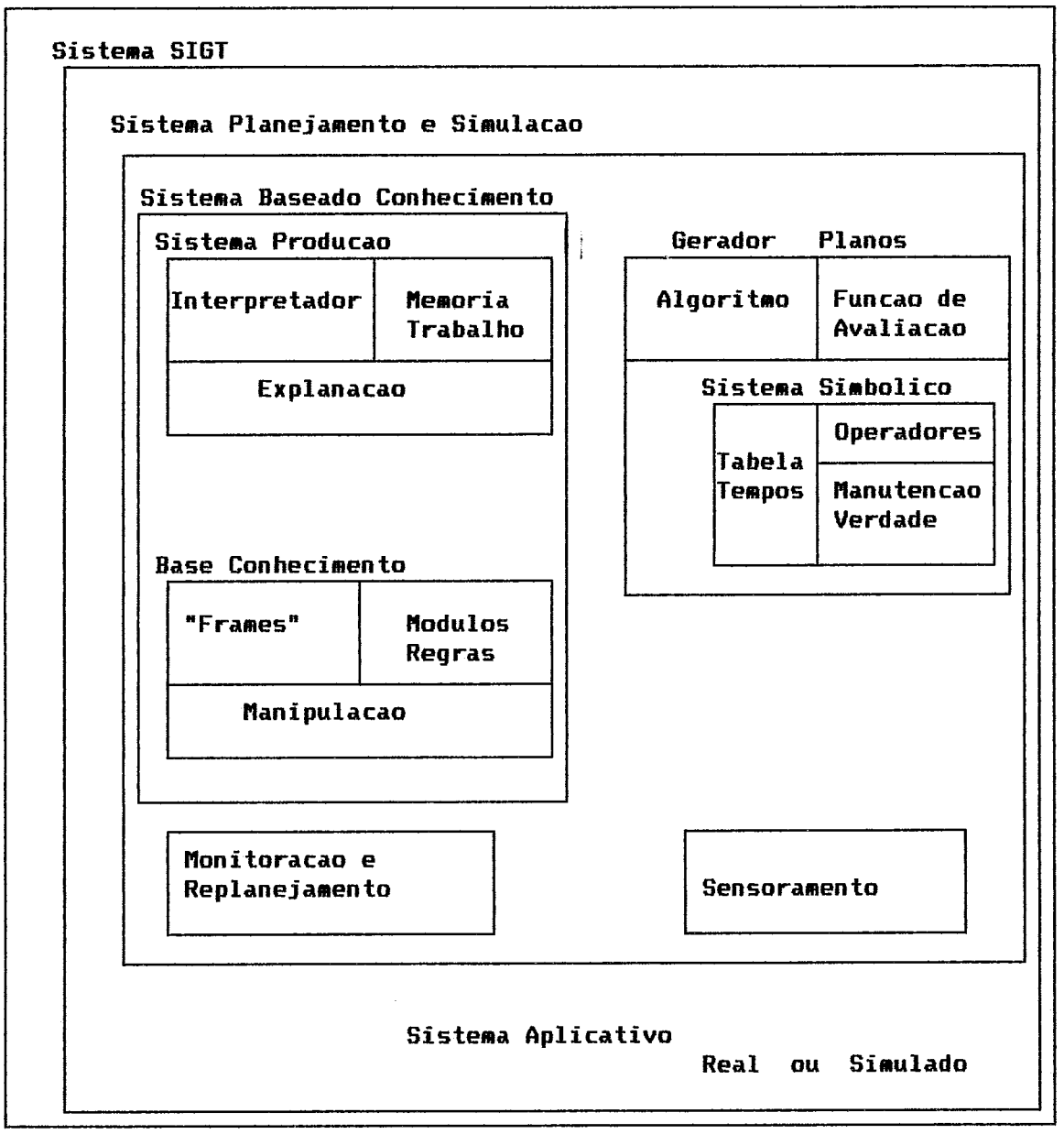


figura 66: organização do sistema SIGT.

Entretanto, quase todos os sistemas aplicativos reais, que exigem a realização de planejamento, não possuem esta característica de extrema previsibilidade. Mesmo se considerassemos a verdade absoluta das previsões ou suposições realizadas ao longo do planejamento, teríamos, como já vimos anteriormente, uma vasta quantidade de problemas envolvidos na tarefa de planejamento, como: "frame problem", explosão

combinatória da busca, manutenção de verdade, representação do tempo, raciocínio temporal, representação de ações, representação do universo de planos, representação e manipulação de restrições, não linearidade de planos, parelelismo entre ações, sinergia, interação entre objetivos, relações de preferência, provas de dominância, etc.

Contudo, muito embora estes problemas sejam de grande dificuldade, existe um consenso entre os planejadores (P. Cheeseman [1988]) de que o principal problema na tarefa de planejamento se deve a existência de certos fatores como: ausência de informações, imprecisão da imformação, imprevisibilidade de parâmetros, pouca confiabilidade das fontes de informação e inferência não monotônica, os quais serão considerados neste trabalho como a realização de planejamento com incerteza.

Uma forma natural de se realizar o planejamento com imprecisão da informação é através do uso de teorias de inferência bayesianas e inferência "fuzzy", como forma de propagar a imprecisão ao longo do planejamento. Na ausência de informações, busca-se o emprego de técnicas de previsão e estimativa de parâmetros, bem como a realização de planos altamente condicionais ou a realização de planos que sejam capazes de reagir com eficiência e flexibilidade as situações de contigências provocadas, possivelmente , por falsas suposições. Neste caso, na existência de inferência não-monotônica é de fundamental importância o uso de sistemas de manutenção de verdade, tanto na fase de planejamento como na fase de execução de planos. Estas questões, bem como a realização de planejamento com incerteza no sistema SIGT serão tratadas com mais ênfase em seções posteriores. Na seção seguinte discutiremos algumas características e parâmetros importantes ao entendimento da dinâmica do planejamento.

7.4.1 - Características e Parâmetros

- Reflexão:

Comforme afirmações de T.A. Linden [1988], apresentadas na seção 7.3.1, podemos considerar a reflexão como um princípio hierárquico que distingue o processo de atuar do processo de raciocinar ou refletir sobre a forma de atuar. Portanto, a existência de ação (execução) em

um nível de abstração envolve, certamente, a existência de reflexão (planejamento) em um nível mais alto.

Imaginando agora, um sistema de controle que atue de forma flexível e inteligente, podemos considerar a existência de uma troca compensativa entre estes dois critérios básicos. Por exemplo, se considerarmos que o processo de reflexão sobre como agir não seja precedido de um planejamento prévio, temos um sistema de alta flexibilidade, alta capacidade de reação, mas, infelizmente, baixa performance quanto a elaboração da resposta e quanto a inteligência ou otimalidade das ações. A respeito da realização de ações inteligentes no sistema SIGT, sob efeito de um planejamento prévio, podemos considerar as ponderações feitas na seção 2.3.4 .

Esta forma de atuação, no caso específico de controle de sistemas dinâmicos com funções de planejamento e "scheduling", seria o exercício de controle com reflexão, por parte do controlador, em resposta a troca de situações com a presenca de uma certa quantidade de informações. Lógicamente, se não houvesse presença de informações, não haveria porque refletir, e a resposta do controle a troca de situações seria imediata, do tipo estímulo-reflexo.

Ao contrário, um sistema de controle sob efeito de planejamento prévio, teria menos flexibilidade ou capacidade de reação a situações de contingência, mas , por outro lado, teria maior performance e maior inteligência. É evidente , que o grau de compensação destes dois critérios se dará em função do grau de incerteza existente no planejamento.

- Previsibilidade e Reatividade:

P.Cheeseman [1988] propõe um diagrama para mostrar a troca compensativa entre a previsibilidade e reatividade em sistemas de planejamento, ver figura 67.

Podemos observar neste diagrama , que em um extremo, como já dissemos, se situa os sistemas de planejamento tradicionais, que consideram no sistema aplicativo a existência de previsibilidade máxima. Neste caso, qualquer alteração da previsibilidade, durante a execução

destes planos, acarreta graves consequências dada a incapacidade de reação a este tipo de contigência.

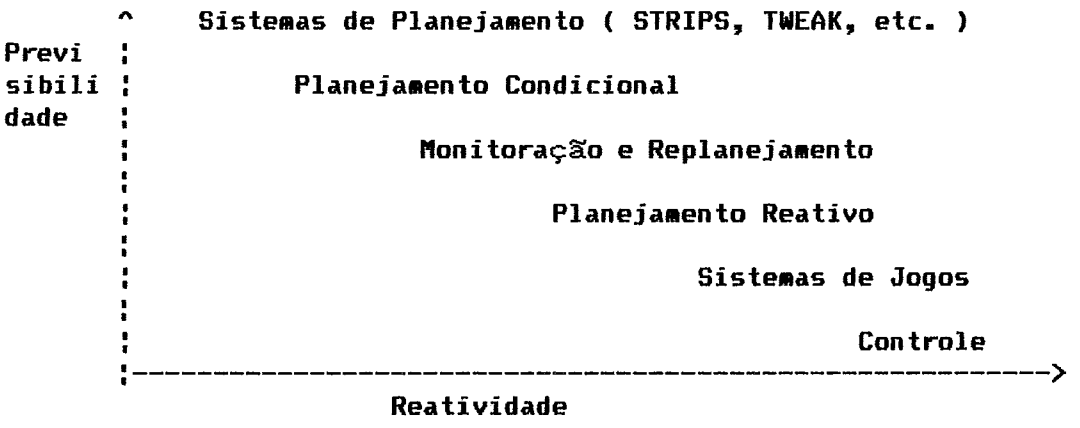


figura 67: diagrama previsibilidade x reatividade

Por outro lado, no extremo oposto do diagrama, temos o emprego da teoria de controle que não possui qualquer forma de planejamento prévio ou avançado. Neste caso, o sistema de controle reage imediatamente as diferenças entre o estado previsto e o estado observado (controlado). Desta forma, o poder de reatividade é total e a previsibilidade do sistema aplicativo é nenhuma. Neste sentido, podemos afirmar que quanto menor a previsibilidade do sistema aplicativo, maior terá que ser a capacidade de reação e flexibilidade do sistema de planejamento e controle.

No caso de sistemas de jogos: xadrez, GO, etc., com grande emprego nos sistemas militares de combate e defesa, o alto grau de reatividade se deve a adversidade do sistema provocada pela incerteza em relação as respostas do adversário e a existência de objetivos conflitantes.

Segundo Cheeseman (na seção 2.3.4 fizemos uma analogia de programas de jogos com o sistema de planejamento do SIGT), a presenca de incerteza em sistemas de planejamento faz com que estes sistemas se comportem, na essência, como sistemas de jogos. Daí, a grande utilidade do desenvolvimento teórico desta área da Inteligência Artificial no desenvolvimento da teoria de planejamento.

- Reatividade e Tempo de Resposta:

É natural considerarmos que quanto mais reativo for um sistema, menor será o tempo disponível para a formulação da resposta. Como situação extrema, podemos considerar o próprio estímulo-reflexo do ser humano.

Neste caso, é importante considerarmos que o parâmetro tempo de resposta não significa medida de performance, como visto anteriormente, ao comentarmos sobre a existência de reflexão no planejamento. Ao contrário, os sistemas altamente reativos devem ter respostas pré-compiladas do tipo estímulo-reflexo, já que o tempo necessário ao raciocínio ou a reflexão no processo de elaboração de respostas é extremamente curto.

Entretanto , podemos concluir que, quanto maior a reatividade do sistema, menor será o tempo disponível para a elaboração da resposta, e menor será a performance do sistema, dado a não existência de planejamento (repostas pré-elaboradas).

- Horizonte de Planejamento e Previsibilidade:

Sem dúvida alguma, ver observações na seção 2.3.1, quanto maior for a previsibilidade do sistema aplicativo maior será o horizonte ou período de planejamento a ser estabelecido. Esta troca, como já vimos, exige a determinação ideal do período de planejamento em função do grau de imprecisão do sistema, de forma a evitar a realização de excesso de planejamento, sendo de vital importância aos sistemas de planejamento em tempo real do tipo monitoração e replanejamento com tempos rígidos de resposta quanto ao reparo ou quanto a reelaboração de planos.

- Reatividade e Planejamento Hierárquico:

Quanto maior o nível das ações na hierarquia de planejamento e controle, mais previsível o sistema aplicativo tende a ser, de modo a reduzir a ocorrência de possíveis falhas que comprometam, sériamente, a realização do planejamento. Como exemplo, no sistema SIGT, a ocorrência de alguma falha, que afete as restrições estruturais do sistema, incorre , a primeira vista , na realização do replanejamento.

Entretanto, se alguma falha compromete uma restrição flexível,

teremos maior controle da situação, bastando, para tanto, reavaliar de forma correta, os intervalos de suporte que determinam a reelaboração das funções de pertinência, no próximo período de planejamento.

- Previsibilidade e Informação Sensorizada:

Claramente, quanto mais reativo for o sistema , maior será a quantidade de informação sensorizada necessária a realização do planejamento/controle, de modo a compensar a imprevisibilidade e reduzir a ausência de informações. Como exemplo, no sistema SIGT, a informação sobre posicionamento de trens, ocupação de pátios e defeitos do sistema, são imprescindíveis para a realização do controle de trens sob planejamento.

- Corretude e Robustez :

Nesta troca, podemos dizer que, tão importante quanto a elaboração de planos corretos nos sistemas mais reativos, é também a realização des planos robustos, que, a exemplo da troca flexibilidade x inteligência , são capazes de reagir a ocorrência de certas contigências dada a imprevisibilidade do sistema. Desta forma, podemos admitir uma maior robustez de planos , e uma certa perda de otimalidade, em benefício de maior flexibilidade, como uma medida satisfatória.

- Enquadramento do Sistema SIGT:

Ao que nos parece, o controle e gerenciamento de trens pode ser colocado como uma forma de planejamento com monitoração e replanejamento , pois o sistema ferroviário apresenta um certo grau de previsibilidade, bem como , exige uma certa capacidade de reação a determinadas mudanças ou contigências, certamente, equânimes.

O tempo necessário a resposta, se não é muito elevado, é suficiente para permitir a elaboração de planos e a realização de planejamento, envolvendo um período de abrangência compatível com o grau de previsibilidade do sistema. Por outro lado, como veremos adiante, é possível ,em alguns sistemas ferroviários, exercer uma forma de controle sobre os parâmetros imprecisos do sistema como, por exemplo, os tempos de percurso dos trens , provocando uma forma de planejamento forçado, na

qual a situação real é que deve se adaptar a situação planejada.

Sem dúvida alguma, achamos que, uma vêz garantida a realização do planejamento com incerteza, o maior problema para se implantar o sistema SIGT em uma situação real de controle, será , certamente, prover ao mesmo capacidade de realizar planejamento em tempo real (aquisição de informações de sensores e capacidade para processamento aproximado), bem como, a realização da tarefa de coordenação de planos consecutivos e de planos adjacentes.

7.4.2 - Manutenção de Verdade em Sistemas de Planejamento

- Lógica e Raciocínio Não-Monotônico:

Nesta seção, veremos como o uso de sistemas de manutenção de verdade podem ser úteis na formação, execução e reparo de planos. Os sistemas de manutenção de verdade permitem a manutenção de registros contendo as suposições feitas durante o planejamento, bem como as justificativas das inferências realizadas de acordo com estas suposições.

Esta forma de inferência que emprega suposições, dado, possivelmente, a ausência de informações ou falta de informações completas, é decorrente de sistemas com raciocínio ou lógica não-monotônica. Neste sistemas, segundo D. McDermott e J. Doyle [1980], um conjunto de fatos ou proposições considerados verdadeiros podem diminuir ou aumentar na presença de novas informações, em consequência das suposições feitas anteriormente, considerando, que, nestes sistemas, e comum a obtenção de conclusões com informações incompletas, na ausência de informações que próvem ao contrário.

Para mostrarmos a complexidade do problema de manutenção de verdade em sistemas com raciocínio monotônico, apresentaremos um exemplo descrito por R. Fikes[1988]:

Nos sistemas monotônicos, pela propriedade de extensão (P.J. Hayes [1973]) ou pela generalização, podemos atestar a seguinte afirmativativa:

" Se garantirmos que, se um fato A for verdadeiro em um instante "to" implica na verdade de um fato B, ou seja: [A,to] ⊃ B, e generalizando p/ ∀ t, então permitimos concluir que:

$$[A,to] \supset [B,to] \Rightarrow [A,t] \supset [B,t]. \qquad (128)$$

Portanto, vimos que nos sistemas monotônicos jamais, na presença de novas informações, revisamos as conclusões obtidas anteriormente.

Entretanto, se uma determinada ação causa a negação de A, em um instante t > to , o que poderíamos afirmar a respeito da verdade de B ?

Pelo raciocínio monotônico, vimos que a verdade de B não se altera.

Pelo raciocínio não-monotônico, considerando a existência de um sistema de manutenção de verdade, sabemos que a verdade de B é garantida pela verdade de A. Portanto, considerando que a verdade de A é falsa em um instante "t" devemos, também, considerar a perda da informação relativa a verdade de B.

Porém, em alguns exemplos, onde se aplica a propriedade de transitividade, a perda de informação pode ser incorreta. Por exemplo:

Se tivermos a implicação:

" Se X está a esquerda de Y e Y está a esquerda de Z então X está a esquerda de Z ",

e considerarmos a movimentação de Y para a direita de Z, pelo sistema de manutenção de verdade, não teremos mais X a esquerda de Z, o que está errado, já que a movimentação de Y não afeta o relacionamento existente entre X e Z.

- Sistemas de Manutenção de Verdade:

Para que um sistema mantenha a verdade dos fatos ou proposições, é necessário que o mesmo mantenha um registro de dependências para que o mesmo possa avaliar, corretamente, no futuro, as consequências de uma suposição errada.

Um dos sistemas mais eficientes voltado para a manutenção de

verdade é descrito por J.Doyle [1979]. Neste sistema, é possível determinar as suposições, na lista de justificativas, nas quais uma conclusão foi baseada, e restabelecer a verdade do conjunto de fatos, através de um esquema eficiente de busca, denominado "backtracking" com dependência direta.

Segundo T. Dean[1988], em sistemas de planejamento onde as informações são organizadas, hierárquicamente, contendo restrições temporais, como exemplo : tabela de tempos, rede procedural ou quadro-negro; para mantermos a verdade dos fatos e, consequentemente, a viabilidade das ações é necessário , a exemplo do sistema de Doyle, desenvolver um mecanismo que permita, de forma eficiente, encontrar as mais recentes suposições que justifiquem, lógicamente, certas conclusões, que possam ser contraditórias, face a introdução de novos fatos. Uma sugestão de T. Dean é no sentido de se organizar os fatos e ações decorrentes do planejamento, segundo o tempo associado, e indexá-las , sintaticamente, de modo a permitir a realização de um "backtracking" eficiente.

Esta filosofia de organização de dados, permite ao sistema SIGT , durante o processo de planejamento, assegurar a verdade da base de conhecimento, e das implicações do sistema de produção, face a constante mudança de estados ao longo das diversas linhas de ação possíveis, no universo de planos realizáveis. Para tanto, guardamos em cada estado, as informações mais recentes que asseguram a justificativa das decisões, como exemplo: informações dos últimos bloqueios decorrentes da solução dos últimos conflitos, incluindo: código da ação, número da seção , código dos trens, tipo de conflito, intervalo de ocupação de pátios ou linhas e instante da ocorrência do conflito.

Estas informações, são guardadas em função da hierarquia de estados ou hierarquia de conflitos, que estabelece uma dependência direta na sequência de ações envolvidas em cada linha do planejamento. Seguindo o exemplo de T. Dean, guardamos, somente, as informações que influenciam, diretamente, na justificativa de novas decisões , descartando as demais.

Desta forma, para a solução de cada novo conflito, no processo de construção de planos, utilizamos estas informações associadas a

alguns procedimentos de inferência, que nos permite expressar o relacionamento entre intervalos de tempo , de modo a restabelecer a verdade da base de conhecimento, possibilitando o uso do sistema de produção , sempre, baseado na verdade atualizada dos fatos. Em seguida, devido a nova decisão, atualizamos o conjunto de informações dependentes, e restabelecemos a condição inicial da base de conhecimento.

No tópico seguinte, veremos como os sistemas de manutenção de verdade podem ser úteis , durante o processo de execução e monitoração de planos, onde os problemas decorrentes da introdução de novas informações, aparecem com grande frequência, dada a imprevisibilidade dos sistemas aplicativos.

- Dependência de Fatos e Planejamento:

Como sabemos, durante a fase de planejamento são feitas várias suposições a respeito de variáveis do sistema resultando na escolha de diferentes caminhos que possam levar ao alcance dos objetivos. Certamente, se as suposições forem falsas , o sistema de planejamento irá falhar na consecução dos objetivos.

No sistema SIGT, podemos considerar como suposições a utilização de valores exatos para a representação de parâmetros nebulosos, ou parametros "fuzzy", relativos aos tempos de percurso dos trens e aos tempos que determinam o início da operação dos mesmos.

Fazemos também , uma suposição a respeito da conduta estacionária do sistema , ou seja, consideramos no processo de planejamento, um conjunto de informações como sendo de natureza estática. Estas informações se referem , na maioria , a descrição de atributos de objetos e entidades do sistema. Entretanto, na fase de realização ou execução do planejamento, podem ocorrer certas transformações que modifiquem , significativamente, os valores dos atributos. Como exemplo, podemos citar a interrupção de um pátio de cruzamento ocasionada por um defeito do sistema de sinalização , ou do sistema de mudança de chave.

Desta forma, durante o processo de execução, na presença de

certas contigências, o sistema de planejamento poderia reparar um plano, recorrendo as suposições que formaram as justificativas associadas a conclusão ou a ação diretamente envolvida na falha.

Neste exemplo, qualquer tentativa de desinterrupção do pátio , que resultasse no restabelecimento da verdade do conjunto de justificativas , implicaria na viabilidade da ação.

Segundo R. Fikes [1988], a maior contribuição que os sistemas de manutenção de verdade poderiam trazer aos sistemas de planejamento, seria fazermos uso da exploração correta das dependências existentes entre as conclusões obtidas durante o processo de inferência e as suposições que levaram a obtenção destas conclusões.

De um modo simplificado, podemos considerar o estabelecimento do plano de ações pelo SIGT, como tal, na medida em que associamos a cada ação os parâmetros essenciais a verificação de sua viabilidade , necessária a execução do plano.

Uma outra sugestão de Fikes seria mantermos a cada passo ou ação do plano uma dependência associada aos objetivos parciais do problema. Neste caso, se, posteriormente, ocorresse a falha de uma ação, seria necessário restabelecer somente os objetivos dependentes. Este processo de planejamento, certamente, reduziria em muito a tarefa de replanejamento ou reparo de planos.

Entretanto, exigiria do problema em questão, algumas características de decomposição. De um modo geral, em problemas que apresentam um grau de interação elevado, esta técnica seria eficiente, somente se afetasse ações comprometidas com aspectos menos importantes do plano, e situadas em um nível mais baixo da hierarquia de planejamento.

Fikes argumenta, também, sobre a possibilidade de se usar informações dependentes como forma de monitorar decisões influenciadas por suposições a respeito de fatos que sejam, posteriormente, de conhecimento do sistema. Desta forma, seria possível atuar diretamente sobre a causa da falha, restabelecendo o planejamento. Por exemplo, se determinado trem se encontrasse, temporáriamente , atrasado, por um motivo qualquer, o sistema poderia autorizar, se possível, um aumento de

velocidade, de modo a recuperar o atraso e garantir a viabilidade do planejamento.

7.4.3 - Representação de Incerteza em Sistemas de Planejamento:

P.R. Bonissone e A.L. Brown Jr. [1986] escreveram um artigo a respeito da realização de planejamento em um domínio complexo na presença de incerteza. Mais precisamente, falam do desenvolvimento de um sistema cuja função maior é permitir o gerenciamento de planos de ações em situações de combate e defesa.

A par das dificuldades de se elaborar um sistema que permita tal planejamento, os autores deram especial atenção a questão relativa a representação de incerteza, descrevendo e qualificando as principais fontes de incerteza, que afetam o processo de raciocínio e inferência e, também, o controle de ações em tais sistemas.

- Fontes de Incerteza:

Os autores destacaram a presença de quatro fontes principais de incerteza:

1- Confiabilidade da informação, afetando, diretamente, o conhecimento factual (fatos) e heurístico (regras) do problema. Neste caso, pode haver a ocorrência de nebulosidade proveniente da utilização de expressões vagas ou quantificadores nebulosos (L.A. Zadeh [1983b]) utilizados para exprimirem o grau de aplicação das regras, a ocorrência de conceitos mal-definidos a respeito de parâmetros observados e, finalmente, a ocorrência de aleatoriedade proveniente da falta de observações suficientes para se tomar evidências , ou da falta de confiabilidade dos instrumentos utilizados para fazer tais observações (sensores);

2 - Imprecisão da linguagem de representação do conhecimento. Este problema tem sido parcialmente estudado pela teoria de raciocínio aproximado na qual os trabalhos na area de raciocínio ou lógica "fuzzy" (J.F. Baldwin [1979] , M.Mizumoto e H.J. Zimmerman [1982] e outros),

tem apresentado grandes contribuições. Neste sentido, não podemos deixar de destacar o emprego de uma regra de inferência composicional (L.A. Zadeh [1983a]) , também conhecida como "modus ponens" generalizado, que permite a existência de um procedimento mais geral de inferência a respeito da existência de predicados "fuzzy" e variáveis linguísticas (L.A. Zadeh [1975]) no antecedente das regras;

3 - Ausência de informação . Neste caso , dada a necessidade de se fazer suposições, temos a ocorrência de uma forma de raciocínio não monotônico, implicando na existência de sistemas de manutenção de verdade, já vistos anteriormente;

4 - Agregação de informações . Neste caso, a ocorrência de incerteza é proveniente da agregação de informações de diferentes fontes de conhecimento ou de diferentes especialistas, em problemas que possuem, em geral, objetivos conflitantes, resultando na ocorrência de contradições ou na redundância de informações na base de conhecimento.

- Modelos de Representação:

De acordo com Bonissone e Brown , os modelos baseados em representação simbólica como os sistemas de raciocínio "default" (R. Reiter[1980]) e sistemas de manutenção de verdade (J.Doyle[1979]) são apropriados para o tratamento da ausência de informações, mas são, geralmente, inadequados para lidarem com informações imprecisas ou pouco confiáveis, devido a falta de valores numéricos, como intervalos de confiança e fatores de certeza, necessários a propagação de incerteza ao longo do processo de raciocínio. Para tanto, são empregados os modelos de representação numérica, cujo objetivo maior esta na formalização de um cálculo, pelo qual seja possível propagar a incerteza da informação ao longo do processo de raciocínio.

Entre as modelos de representação numérica , os autores destacam os modelos Bayesianos, tendo como principal desenvolvimento o modelo de raciocínio inexato de B.G. Buchanan e E.H. Shortlife [1984], a teoria de evidência de G. Shafer [1976] e A.P. Dempster [1967] e, finalmente, a lógica "fuzzy" que inclui a teoria de possibilidade e necessidade de L.A. Zadeh [1983a],[1979].

- Incerteza no Sistema SIGT:

No sistema SIGT tratamos, principalmente, com dois tipos de incerteza: imprecisão de parâmetros ou de valores numéricos e ausência de informações devido a imprevisibilidade do sistema. Quanto ao primeiro tipo de incerteza, associamos , nos modelos de simulação, um modelo de lógica "fuzzy" apresentado na seção 3.6.1, quando consideramos a existência de parâmetros "fuzzy" no sistema.

Quanto a ausência de informações, cuja incerteza se reflete durante a execução de planos, consideramos, como já foi dito, o desenvolvimento de um sistema de manutenção de verdade que permite ao sistema atestar a viabilidade de cada passo do planejamento através da verificação das suposições feitas durante a fase de elaboração dos plano, face ao aparecimento de novas informações.

Neste sentido, a viabilidade de cada passo do planejamento é minuciosamente atestada através da realização de inferências no sistema de produção, que , a exemplo do modelo de simulação, possui a sua informação (base de conhecimento) atualizada dinâmicamente, ou seja, em tempo real. Neste caso, se durante a fase de execução, alguma nova informação causar a inviabilidade de uma ação do planejamento, o sistema de planejamento rejeitará a ação, através da realização de inferência, e determinará a rotina de execução as providências adequadas a realização do replanejamento ou ao reparo do plano.

7.4.4 - Lógica Não-Monotônica e Teoria de Decisão

Nesta seção, queremos destacar o trabalho de C.P. Langlotz e E.H. Shortlife , onde foi discutido a utilização de lógica não-monotônica e o emprego de teoria de decisão como alternativas de metodologia para a realização de planejamento com incerteza.

- Inferência Não-Monotônica e Preferências:

J. Doyle [1985] e Y. Shoham [1987] expressam a idéia de que a realização de inferências não-monotônicas em regras de conhecimento podem ser interpretadas como manifestações de preferências. Shoham ,

que formalmente define uma lógica de preferências em seu trabalho, argumenta que as várias formalizações de inferências não monotônicas são diferentes métodos no sentido de operacionalizar preferências de algumas inferências sobre outras.

Desta forma, segundo Langlotz e Shortlife, o relacionamento entre inferência não-monotônica e teorias de probabilidade, ver figura 68, não é totalmente suficiente, já que este modelo deve ser extendido no sentido de capturar a noção de utilidade, essencial para a determinação de preferências.

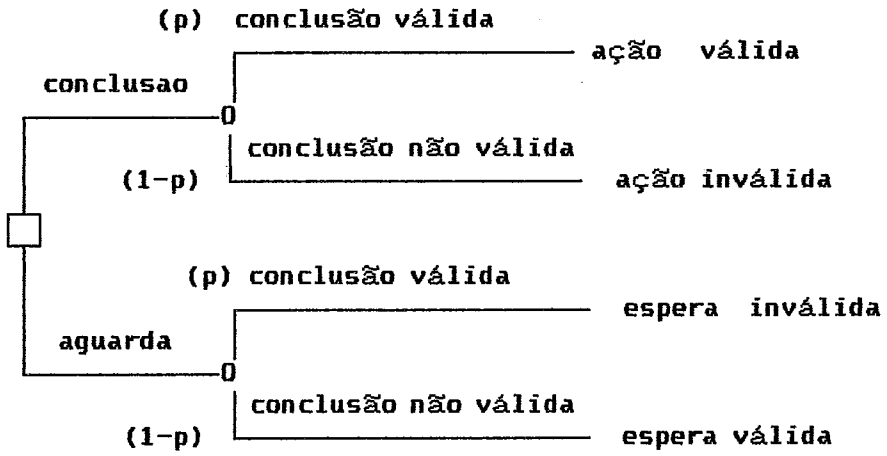


figura 68: árvore de decisão que representa as possibilidades de se tomar uma conclusão não-monotônica em função da probabilidade p.

Para Shoham , existem várias formas de determinamos uma razão que estabeleça uma preferência em determinada inferência. Por exemplo, inferências que são provavelmente válidas são preferiveis sobre inferências, provavelmente, menos válidas. Neste sentido , Shoham afirma que, ao realizarmos uma inferência não-monotônica , devemos maximizar uma utilidade esperada.

O princípio geral que associa a noção de utilidade a realização de inferências não-monotônicas, observa a relação de que, quanto mais séria for a consequência de uma conclusão não-monotônica inválida, mais alto será o valor da probabilidade de realizarmos uma inferência correta. Consequentemente, a probabilidade de realizarmos uma inferência errada decresce. Desta forma , duas conclusões não monotônicas com mesmas probabilidades de erro , podem não serem

indiferentes se a consequência do erro de uma for grave e da outra insignificante.

Portanto, a validade de uma conclusão não-monotônica dependerá , em parte, da desejabilidade ou não do efeito de uma conclusão incorreta.

- Lógica Não-Monotônica e Planejamento:

O apelo ao uso de lógica não-monotônica em sistemas de planejamento é, em grande parte, devido a possibilidade de desenvolvermos uma forma de representação independente do domínio ou do problema em questão. Entretanto, vimos anteriormente, que a válidade de uma conclusão não-monotônica é dependente de outras conclusões da base de conhecimento , produzindo uma interdependência em relação aos objetivos do problema.

Ou seja, considerando que as conclusões não-monotônicas são produzidas e ordenadas de modo a refletir determinadas situações de planejamento com valores específicos de probabilidade, dizemos que a válidade destas conclusões são intrínsecamente dependentes dos objetivos do problema .

Portanto, segundo conclusão dos autores, o uso de lógica não-monotônica seria mais apropriado em problemas onde as utilidades relatadas as consequências das conclusões são menos importantes, ou igualmente importantes, retratando uma forma de indiferença.

Convém relembrar, que a utilização de medidas de preferência ou de utilidade , que são fundamentais no processo de inferência na tarefa de construção de planos, não são, explícitamente, representadas no formalismo da lógica não-monotônica (D. McDermott e J. Doyle [1980]). Entretanto, mesmo se esta representação fosse possível, o uso de preferências em função de novas situações do problema , comprometeria a questão de independência requerida nos modelos de lógica.

- Teoria de Decisão e Planejamento:

Em contraste com a lógica não-monotônica , a teoria de decisão emprega um conjunto de probabilidades e medidas de utilidade associadas

aos objetivos das ações de planejamento. Desta forma , uma decisão , ou um conjunto de ações específicas de um plano, possuem uma utilidade esperada definida por :

$$EU(P_i) = \sum_{\_\_} p(O_j / P_i).\ U(O_j) , \qquad (129)$$

onde:

$EU(P_i)$ se refere a utilidade esperada do i-ésimo plano ;

$p(O_j / P_i)$ se refere a probabilidade de se alcançar a situação O_j após a execução do plano P_i , e

$U(O_j)$ se refere a utilidade associada a situação O_j .

Lógicamente, o plano com maior utilidade esperada será considerado o plano dominante ou superior. Certamente, a grande dificuldade desta técnica de planejamento está na determinação dos valores de utilidade e probabilidade e, também, na capacidade de restrição do conjunto de possiveis planos a um conjunto de planos tratável, ou seja, especificar eficientemente a prova de dominância , bem como as relações de preferência entre planos.

Ademais, o procedimento de inferência utilizado pela teoria de decisão , baseado no uso da fórmula de utilidade esperada, nem sempre corresponde as técnicas de inferência empregadas pelo raciocínio humano em problemas de planejamento ou tomada de decisão sob incerteza.

A respeito desta última observação, queremos destacar que o procedimento de inferência do sistema SIGT se baseia na teoria "fuzzy" de tomada de decisões (R.E. Bellman e L.A. Zadeh [1974]) , permitindo a seleção de planos, segundo os critérios apresentados no capítulo III. No entanto, em nosso procedimento, não fizemos uso de probabilidades de modo a avaliar o efeito das ações ao longo do planejamento, mas somente de funções de utilidade representadas por um cojunto de funções de pertinência na forma conjuntiva.

O uso de probabilidades seria interessante , no sentido de avaliar tão quanto o estado ou situação de saída se assemelha ao estado ou situação desejável. Neste ponto, consideramos o sistema SIGT como um

sistema de planejamento de linha estratégico, que realiza uma previsão rigorosa , admitindo um conhecimento absoluto, a respeito do efeito das ações que serão, posteriormente , executadas.

No trabalho de T.A. Linder e J. Glicksman [1988] , descrito posteriormente, é proposto o desenvolvimento de um sistema de planejamento sob incerteza, que faz uso de um algoritmo A* com função de custo probabilística, de modo a evitar possíveis dependências sobre situações consideradas incertas e , ao mesmo tempo, críticas para o sucesso do planejamento.

Desta forma, podemos nos referir a este tipo de procedimento como uma forma de penalização da incerteza, visando, essencialmente , a produção de planos robustos em sacrifício da otimalidade. Como veremos adiante, a possibilidade da introdução de uma função de custo probabilística em nosso sistema de planejamento , é considerada , a primeira vista, viável, visando, a exemplo do sistema de Linder e Glicksman, a penalização de situações que comprometam, em função da incerteza, a viabilidade do planejamento.

7.5 - Planejamento Temporal

Segundo A.A. Haas [1985], todas as situações decorrentes dos sistemas de planejamento anteriormente discutidos, são meras possibilidades baseadas na representação de estados e eventos na forma do cálculo de situações (J. MacCarthy e P.J. Hayes [1965]), que estabele uma situação como um intervalo de tempo no qual não se realizam alterações ou não se invalidam as proposições carcterísticas de um estado.

Estes sistemas, na maioria, se preocupam em elaborar planos estabelecendo, préviamente, um conjunto de situações desejáveis, bem como o efeito de possíveis ações, não se preocupando com os problemas relacionados a futura execução do planejamento. Entretanto , um sistema de planejamento que se preocupe, conjuntamente , com a tarefa de planejamento e com a viabilidade da execução de planos, deve ser capaz de raciocinar sobre futuros eventos bem como sobre os eventos atuais que incluem , óbviamente, situações e ações.

Para A.A. Haas [1985], uma forma eficiente de incluir situações atuais em um sistema de planejamento, é através da introdução de intervalos de tempos associados a um conjunto de predicados que possibilitam a realização de inferências a respeito da ocorrência de situações. Neste sentido, veremos nas seções seguintes como o nosso sistema trata a questão de representação de tempo e ações, bem como a realização de raciocínio temporal sobre restrições características do problema de sequenciamento de trens.

7.5.1 - Estruturas de Representação

- Situações como Instantes ou Intervalo de Tempo:

Segundo E. Charniak e D. McDermott [1985], existem duas formas mais comuns de representação de tempo e situações na forma de redes associativas. A primeira, representa o início e fim de atividades como nós ou estados, e intervalos de tempos em arcos ou transições. Esta forma de representação trata a situação como característica de instantes ou pontos de tempo, e admite formas de relacionamento mais

simples. A segunda , representa as atividades ou situações como nós ou estados, e os arcos ou transições como relações temporais entre situações. Neste caso, uma situação é tratada como característica de intervalos de tempo, e as formas de relacionamento são mais complexas.

Seja, por exemplo , o plano de ações definido na seção 7.3.2, para o problema de conflito de trens, segundo a situação inicial, descrita pela figura 69.

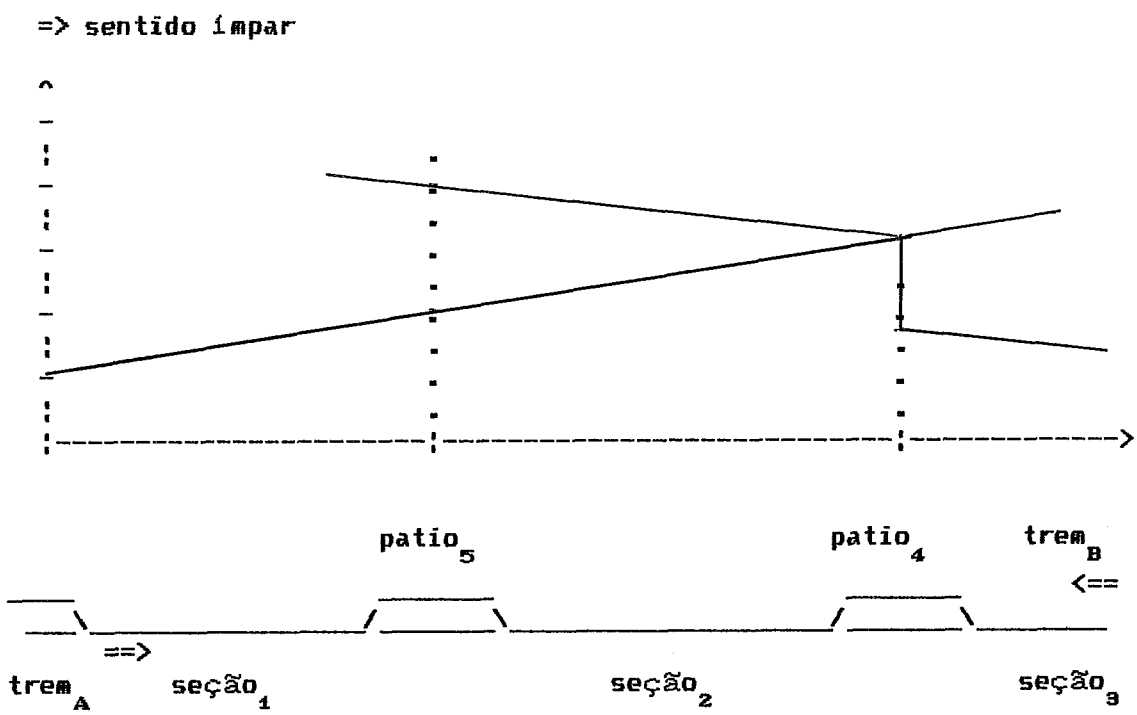


figura 69: conflito de trens.

plano de ações:

AV (A,2), DESV (B,4), AV (A,3), AV (B,2), AV(B,1).

A execução destas ações na forma sequencial, estabelece um conjunto de situações que pode ser representado em uma rede associativa, tratando situações como instantes ou como intervalos de tempo, ou seja, segundo as figuras 70 ou 71.

Baseado nestas representações, podemos observar que a representação em intervalos de tempo, possui menos estados e formas de relacionamento mais complexas exigindo, portanto, procedimentos de inferência mais complexos. Por outro lado, a representação em instantes de tempo possui apenas dois tipos de relacionamento: antes e igual,

haja visto que os estados representam, explícitamente, o início e fim das atividades.

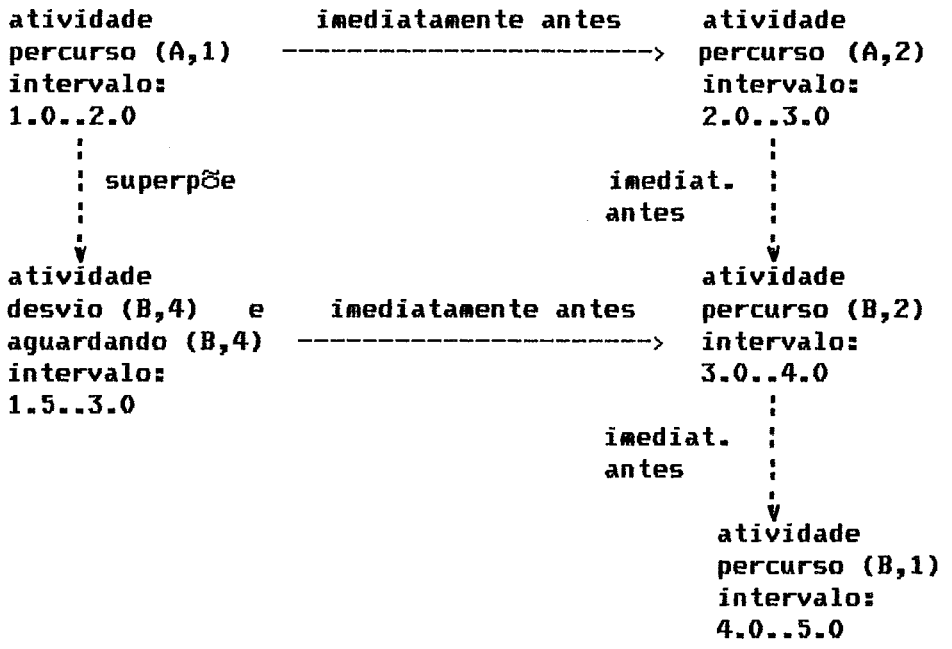


figura 70: situações como intervalos de tempo.

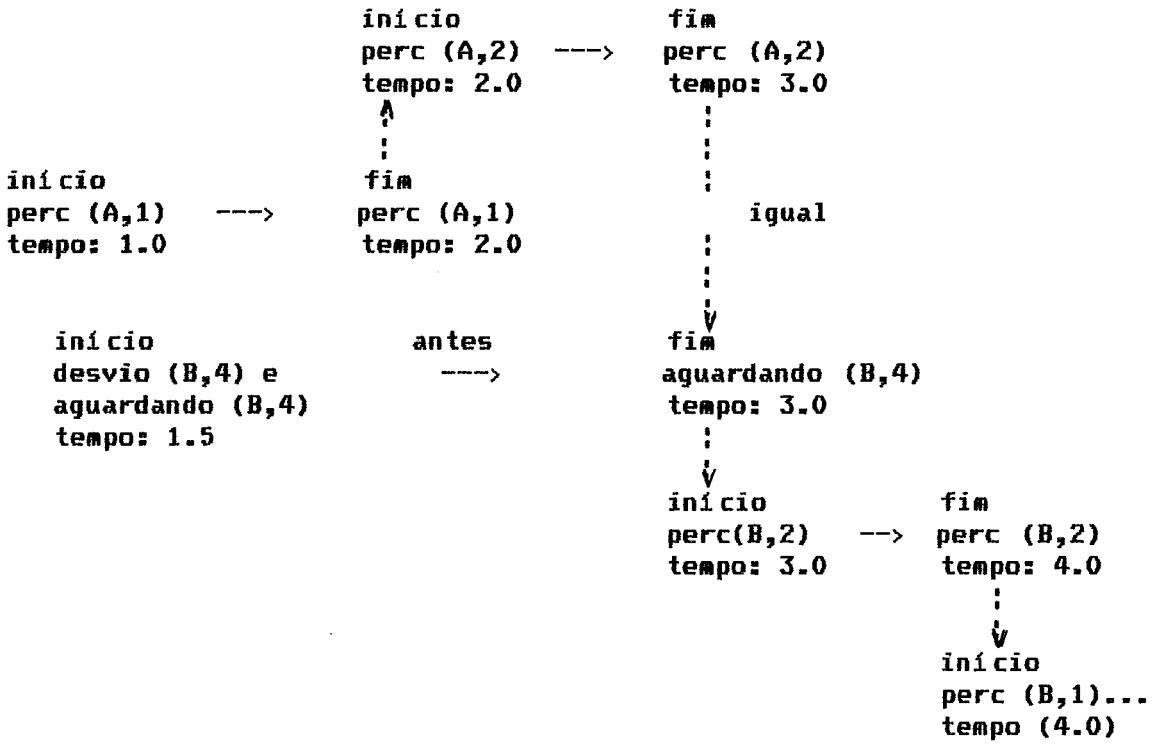


figura 71: situações como instantes de tempo.

Também, podemos considerar a representação em intervalos de tempo, como relacionada a modelagem de atividades segundo a realização

de simulação por processos, enquanto que, a representação em instantes de tempo, como característica da modelagem tradicional da simulação por eventos, que estão associados ao início e término das atividades, ver seções 6.4.1 e 6.4.2 .

Para o sistema SIGT consideramos a utilização de uma tabela para representar os instantes de tempo característicos do início das atividades, uma vez que a duração das mesmas é de nosso conhecimento. Ao longo do processo de planejamento, realizamos uma forma de raciocínio sobre intervalos de tempos, utilizando um conjunto de funções sobre cálculo de intervalos que operam a partir dos dados fornecidos pela tabela . Estas informações, são suficientes para raciocinarmos sobre as restrições temporais que envolvem as ações de sequenciamento, bem como a solução de conflitos , durante a elaboração de planos.

- Raciocínio Temporal no Sistema SIGT:

Entretanto, se considerarmos em nosso planejamento, a seguinte questão: na execução do passo DESV(B,4), para que o mesmo seja viável, é necessário que haja capacidade suficiente na posição 4, ou , em outras palavras, que o pátio de cruzamento esteja desocupado. Na certa, seria por demais displicente, que nosso sistema não raciocinasse sobre possíveis ocupações neste patio, ou desconsiderasse as restrições de recurso do problema.

Infelizmente, para tanto, a realização de inferências na tabela de tempo, que representa informações sobre as restrições de sequenciamento (eventos passados e futuros) e restrições de precedência (conflito atual, e conflitos anteriormente resolvidos) , não é suficiente para resolver este tipo de problema.

Para tanto, como já dito em seções anteriores, fizemos uso das informações dependentes que resultaram na proposta de execução do passo, ou seja, das infomações decorrentes da resolução do conflito entre os trens A e B na seção$_2$, representadas no estado presente do processo de busca.

Neste estado , obtemos as informações do conflito em questão, bem como das consequências da ação ou ações anteriores que estabeleceram a dependência direta em relação a nova ação proposta (neste caso

estamos nos referindo a existência de um "backtracking" por dependência direta entre os vários conflitos dentro de uma mesma linha de ação).

Para que a ocupação do pátio de cruzamento em um conflito em linha singela seja verificado, realizamos o seguinte procedimento:

1 - Verificamos se a seção presente que contém o pátio de cruzamento é a mesma referida nas informações dependentes. Esta informação dependente é colocada sob a forma de bloqueios caracterizados pelo número da seção ocupada, pelo tipo de conflito, pelo tipo de ação executada , pelo código do trem e pelo intervalo de ocupação ou janela;

2 - Caso afirmativo, em função do tipo do conflito e da ação realizada na informação dependente, verificamos qual pátio de cruzamento foi ocupado anteriormente (lembrando que uma seção singela possue dois pátios de cruzamento). No nosso exemplo estamos nos referindo a viabilidade de cruzamento no $pátio_4$;

3 - Em seguida, verificamos se o trem envolvido na ação presente não é o mesmo trem envolvido na informação dependente (caso fosse , teríamos a situação na qual um trem já aguarda em um pátio de cruzamento e deve continuar aguardando, em função da resolução de dois conflitos consecutivos, possivelmente, dado a realização de cruzamento com uma onda de trens);

4 - Caso afirmativo , calculamos o intervalo de ocupação do pátio de cruzamento como consequência da execução da ação presente e da ação oposta (se pararmos o $trem_A$ no $pátio_5$, no nosso exemplo, o intervalo será de 1.5 à 3.0), e extraímos , da informação dependente, a janela ou intervalo de ocupação do mesmo pátio em função das ações anteriores;

5 - Verificamos se ocorre a interseção de intervalos caracterizada pelas funções de superposição, igualdade, ou inclusão dos intervalos em questão;

6 - Caso afirmativo, o sistema determina a atualização da capacidade do pátio de cruzamento da seção presente, segundo os dois tipos de ações , dado o tipo de conflito. Neste caso, decrescemos a capacidade do patio 4 ou do 5 em uma unidade. Estas informações são, essencialmente,"slots" de "frames" que compõem a base de conhecimento do

sistema;

7 - Uma vez atualizada a informação de capacidade, o sistema de produção verifica a viabilidade ou não de se executar a ação em questão, através da determinação de preferências. Por exemplo , se não houvesse capacidade no pátio$_4$, teríamos uma preferência para parar o trem$_A$ no pátio , e, finalmente,

8 - Garantida a execução da ação, o sistema atualiza no estado presente as informações dependentes, informando a ocorrência de novo bloqueio, e restaura (apaga) as informações gravadas, anteriormente,na "frame" da base de conhecimento.

Desta forma, permitimos ao sistema de produção trabalhar de uma forma semelhante ao cálculo de situações, ou seja, raciocinando sempre sobre fatos verdadeiros condicionados a situações atuais. Este procedimento, caracterizado na organização do SIGT como um sistema de manutenção de verdade, permite ao planejamento:

1 - Verificar a viabilidade de ações quanto a cruzamentos em linha singela (exemplo anterior);

2 - Verificar a viabilidade de ações quanto a ultrapassagens em linha singela;

3 - Verificar se algum trem já aguarda em cruzamento;

4 - Verificar se algum trem já aguarda em ultrapassagem;

5 - Verificar a formação de onda de trens;

6 - Verificar a viabilidade de ações quanto a cruzamentos em linha dupla;

7 - Verificar a viabilidade de ações quanto a ultrapassagens em linha dupla.

Neste sentido, observamos que o nosso sistema de planejamento , baseado na representação de situações e atividades e na realização de operações de relacionamento entre intervalos , adquiriu capacidade suficiente para raciocinar sobre as restrições temporais de sequenciamento, precedência e recursos do problema de sequenciamento de trens. Os predicados utilizados para o cálculo de relacionamento entre intervalos foram desenvolvidos, principalmente, segundo os trabalhos de J.F. Allen [1983] e [1984], os quais serao descritos a seguir.

7.5.2 - Teorias de Tempo e Ação

- Raciocínio sobre Intervalos de Tempo:

J.F. Allen [1983] apresenta uma lógica temporal baseada na concepção de intervalos de tempo como primitiva para a representação do conhecimento temporal . Allen apresenta, tambem, um algoritmo de raciocínio temporal baseado na técnica de propagação de restrições que permite manter as relações temporais entre um conjunto de fatos dada a inclusão de novas restrições.

Segundo Allen , a representação temporal baseada na técnica espaço-estados , a exemplo do sistema STRIPS, caracteriza uma ação como uma função de mapeamento entre dois estados consecutivos em instantes distintos de tempo. Desta forma, chega-se a uma representação do tempo , considerando a descrição dos fatos ao longo dos sucessivos estados.

Entretanto, dado ao excessivo gasto de memória, muitos sistemas mantém as informações , sómente, do último ou dos últimos estados , preservando as proposições verdadeiras e deletando as proposições que se tornam falsas. Esta visão de representação de tempo e ação , característica dos sistemas de solução de problemas , foi muito influenciada pelo trabalho de J. MacCarthy e P. J. Hayes [1969] sobre o cálculo de situações, sendo a base de muitos sistemas de planejamento de linha estratégica.

Uma nova forma de representação de tempo e ação, proposta por Allen, seria através do uso de intervalos. Neste caso, ações e eventos , ao invés de funções de mapeamento, seriam considerados como conjuntos de intervalos ao qual os mesmos pertenceriam, considerando a possibilidade de ações instantâneas e ações duradouras.

D. McDermott [1982] também propos uma lógica temporal baseada no uso de intervalos de tempo como primitiva, considerando a segunda forma de representação da seção 7.5.2, ou seja, definindo situações ou estados como instantes de tempo, e intervalos como transições entre estados.

A respeito da descrição de ações , J. Allen[1984] considera a

utilização de descrições funcionais com argumentos, enquanto que, McDermott [1982], considera o uso de expressões ou fórmulas do cálculo de predicados. Segundo L. Morgenstern [1988], as vantagens de uma representação sobre a outra são circunstânciais, no sentido de que, a descrição de McDermott só é mais expressiva para a representação de ações compostas, as quais não podem ser descritas na forma funcional.

- Teoria Pontual e Teoria de Intervalos:

Considerando a definição de eventos ou ações como um instante de tempo decorrente da transição entre estados , não permitimos maior flexibilidade ao planejamento quanto a realização de raciocínio sobre futuras possibilidades. Ao contrário, considerando ações ou eventos como intervalos de tempos ou atividades (segundo a primeira forma de representação da seção 7.5.2), tratamos sempre da possibilidade de uma transição entre estados se efetuar ao longo de um intervalo, tornando mais flexível o planejamento sobre futuros eventos ou ações.

Da mesma forma , podemos considerar que proposições são verdadeiras ou falsas durante períodos ou intervalos , melhor do que em pontos ou instantes do tempo.

Dada a existência de intervalos de tempos, caracterizando as atividades ou ações (por exemplo, no sistema SIGT a ação AV(A,2) só se completa quando o $trem_A$ termina a atividade de percurso na $secao_2$), surgem um número significativo de predicados de modo a definir o relacionamento entre estados. Para representar estas relações , J. Allen[1983] faz uso de pontos ou instantes de tempo para caracterizar o início e o fim de intervalos.

Entretanto, para que não ocorra situações contraditórias em um único instante de tempo, e, também, para que não ocorram instantes de tempo descaracterizados de uma situação, Allen propõe o uso de intervalos fechados em um único ponto.

Desta forma, considerando "s" como o ponto inicial e "t" como o ponto final dos intervalos X e Y, temos as seguintes relações entre intervalos, descritas por Allen[1983], considerando algumas alterações na simbologia, conforme mostra o quadro 7.

quadro 7: relação entre intervalos:

predicado	relação	condição
BEF(X,Y) AFT(X,Y)	X antes de Y ou Y depois de X	X.t < Y.s
EQU(X,Y)	X igual a Y ou Y igual a X	X.s = Y.s e X.t = Y.t
MEET(X,Y)	X imediat. antes de Y ou Y imediat. depois de X	X.t = Y.s
DUR(X,Y)	X durante Y ou Y contém X	X.s > Y.s e X.t < Y.t
STA(X,Y)	X inicia com Y	X.s = Y.s e X.t < Y.t
FIN(X,Y)	X termina com Y	X.s > Y.s e X.t = Y.t
OVE(X,Y)	X superpõe Y ou Y é superposto por X	Y.s < X.t e X.t < Y.t e Y.s > X.s

Obviamente, se fizermos em algum intervalo, coincidirem os pontos de ínicio e fim , teremos um único instante de tempo denominado "i", ou seja:

X.t = X.s = i
Y.t = Y.s = i

- Modelo de Allen:

J.F.Allen [1984] apresenta um formalismo para o raciocínio sobre ações, baseado no uso das primitivas de relacionamento entre intervalos, descritas anteriormente. A teoria proposta por Allen inclui a descrição de aspectos dinâmicos e estáticos de um domínio , utilizando uma linguagem de cálculo de predicados de primeira ordem.

Baseado nas primitivas apresentadas, Allen descreve o predicado HOLDS(p,T) associado a idéia de situações , ou seja, de que uma propriedade ou proposição "p" se mantem verdadeira durante o intervalo de tempo T.

Entretanto,para que "p" seja verdadeiro durante todo o intervalo T, é necessário que "p" seja verdadeiro em todo sub-intervalo de T. Para tanto, Allen define o predicado :

IN(t,T) <=> (DUR(t,T) OU STA(t1,T) OU FIN(t,T)) , (130)

derivando:

HOLDS(p,T) <=> (∀ t. IN(t,T) => HOLDS(p,t)). (131)

Em alguns casos, certas proposições , como exemplo um evento, podem ocorrer em algum sub-intervalo de T, mas não em todos sub-intervalos, tornando proibitivo o uso do predicado HOLDS. Para tanto, Allen define o predicado OCCUR associado a idéia de ocorrência . Portanto, o predicado OCCUR será verdadeiro, sómente se o evento acontecer sobre algum intervalo "t", e não existir nenhum outro sub-intervalo de "t", sobre o qual o mesmo tenha ocorrência, derivando o seguinte axioma:

OCCUR(e,t) E IN(t',t) => NÃO (OCCUR(e,t')) (132)

Segundo Allen , a definição de ação ocorre , normalmente, associando um agente a ocorrência de um evento, ou seja:

ACAUSE(agente, ocorrência) (133)

Esta ocorrência pode ser do tipo anterior, ou seja: OCCUR(e,t) , ou do tipo OCCURRING, associada a existência de processos ou atividades. Portanto, podemos ter o novo axioma:

OCURRING(p,t) => ∃ t'.IN(t',t) E OCURRING(p,t') (134)

Ou seja, se um processo tem duração sobre um intervalo "t", o mesmo deve acontecer sobre cada sub-intervalo de "t". Lógicamente, um processo só será contínuo para todo intervalo, se não houver a ocorrência de interrupções.

- Trabalhos Correlatos:

J.F. Allen e P.J. Hayes [1985] mostram que as relações temporais primitivas entre intervalos ou períodos , podem ser definidas em termo do predicado MEET, atravás do desenvolvimento de cinco axiomas básicos.

Posteriormente, E.P.Tsang [1988] descreve o desenvolvimento de

uma lógica temporal , tendo como primitiva a noção de evento , sendo totalmente compatível com a teoria de intervalos de Allen e Hayes. Mais precisamente, Tsang propõe uma estrutura de eventos desenvolvida com base nos conceitos de instantes ou pontos de tempo, argumentando que cada teoria pode ser construída segundo a outra e vice-versa. Desta forma, como vimos no quadro 7 , através do uso de pontos definindo intervalos de tempo, é possível expressarmos qualquer relação temporal descritas por Allen, através do uso de um conjunto de inequações na forma conjuntiva.

Segundo Tsang, o uso de um cálculo de intervalos baseado em instantes de tempo é necessário em sistemas de planejamento onde as informações existentes se referem ao início e fim das atividades , a exemplo de nosso sistema. Ademais, as operações entre instantes de tempo envolve somente o uso dos três operadores básicos: maior, menor e igual.

É possível, também, ao trabalharmos com pontos de tempo , exercemos um controle mais flexível sobre a granularidade do tempo, ao representarmos intervalos como um conjunto de pontos distribuídos de forma densa em uma linha de valores reais, com determinado número de casas decimais. Em nosso sistema, por exemplo, consideramos a representação interna em ponto flutuante para a realização do cálculo de intervalos . Todavia, na visualização de valores de tempo, é considerada a conversão para a representação digital horária, com precisão de centésimos de segundo.

- Extensão ao Modelo de Allen:

Paralelamente ao trabalho de Tsang, P.J. Hayes e J.F. Allen[1988] apresentam uma extensão a teoria, incorporando algumas discussões a respeito da existência de pontos e momentos de tempo. Neste trabalho, os autores tambem confirmam a equidade da teoria de pontos de tempos em relação a teoria de intervalos, considerando pontos como locais abstratos nos quais períodos ou intervalos se encontram . A extensão proposta pelos autores permite a teoria manter o raciocínio sobre intervalos, considerando, entretanto, maior flexibilidade quanto as suposições de granularidade feitas tempo a tempo.

Neste sentido, a nova representação de encontro de períodos é compatível com modelos densos, nos quais períodos se encontram segundo intervalos abertos; como também , com modelos discretos, segundo a construção de intervalos como conjunto de pontos de tempos unidos ou não pela presença de "singletons" que caracterizam momentos.

- Cálculo de Intervalos no Sistema SIGT:

Voltando ao procedimento de verificação da seção 7.5.1 , onde, no passo 5 , analizamos a existência de interseção entre um conjunto de intervalos; se considerarmos como T o intervalo de bloqueio da informação dependente e P o intervalo durante o qual o trem aguardará no pátio de cruzamento, teremos que verificar as seguintes relações, que provocam a interseção de intervalos:

EQU(T,P), STA(T,P) , STA(P,T), FIN(T,P), FIN(P,T), DUR(T,P), DUR(P,T), OVE(P,T) e OVE(T,P),

ou, considerando o predicado IN(t,T):

EQU(T,P), IN(P,T), IN(T,P), OVE(P,T) e OVE(T,P). (135)

Obs: o predicado MEET(X,Y) implica, pela teoria de intervalos, na inexistência de interseção.

Entretanto, como a verificação destas relações resultará em um cálculo bastante complexo, desenvolvemos um novo procedimento de verificação de interseções, baseado no fato de que:

1 - Se considerarmos o instante de ocorrência do conflito como "p", teremos, necessariamente, a verdade do predicado HOLDS(p,P) (observe que, como situação extrema , P = p , acarretando a solução imediata do conflito com a realização de cruzamento em movimento);

2 - Se considerarmos a existência do predicado HOLDS(p,T), teremos:

Se HOLDS(p,T) for verdadeiro, então ocorre a interseção dos intervalos P e T, já que HOLDS(p,P) também é verdadeiro.

Caso contrário, ou seja HOLDS(NÃO.p,T) => NÃO.HOLDS(p,T), teremos duas possibilidades, de acordo com a figura 72.

2.1 - p > T.t , ou seja, o instante do conflito se verifica acima do ponto superior do intervalo T;

2.2 - p < T.s, ou seja, o instante do conflito se verifica abaixo do ponto inferior do intervalo T.

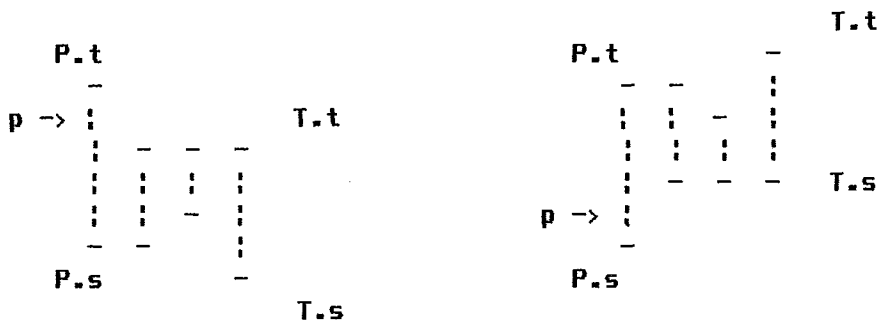


figura 72: relações entre P e T dado que NÃO.HOLDS(p,T).

Portanto, para verificarmos a interseção entre os dois intervalos P e T, devemos verificar as seguintes condições:

HOLDS(p,T) ou

NÃO.HOLDS(p,T) e (p > T.t) e ((P.s = T.s) ou (T.s > P.s) ou (T.s < P.s e T.t > P.s)) ou

NÃO.HOLDS(p,T) e (p < T.s) e ((T.t = P.t) ou (T.t < P.t) ou (T.t > P.t e T.s < P.t))

Considerando, a representação equivalente do predicado HOLDS na representação de pontos e, simplificando a expressão acima, teremos :

(p ≤ T.t) e (p ≥ T.s) ou
(p > T.t) e (T.t > P.s) ou (136)
(p < T.s) e (T.s < P.t),

reduzindo, enormemente, o cálculo necessário a realização de inferências a respeito da verificação da interseção de intervalos.

7.5.3 - Verdades e Planos

- Mundo Esperado, Desejado e Planejado:

J.F. Allen [1984] , a exemplo de A. Haas [1982], descreve a verdade como um predicado cujo objetivo é expressar o fato de que um

agente acredita na verdade de uma proposição. Desta forma o predicado:

HOLDS(BELIEVE(agente,HOLDS(p,Tp),T) <=> (137)
BELIEVE(agente,p,Tp,T),

sugere que um agente acredita durante o intervalo T, que a proposição "p" permanecerá verdadeira durante o intervalo Tp.

Segundo Allen, a definição de planos como uma sequência ou ordenamento parcial de ações, acreditadas pelo agente, é ideal para os sistemas de planejamento de linha estratégica. Contudo, a realização de planejamento em certos domínios , podem alterar de forma significativa o planejamento inicial em relação as intenções do agente, no que tange a execução das ações de planejamento, em decorrência da interferência de certas contigências, provocadas pela ocorrência de eventos externos, ou pela presença de outros agentes.

Por exemplo, no nosso sistema de planejamento, a retirada de um trem de circulação, durante a execução do plano, causa a suspensão imediata de um número finito de conflitos, alterando a verdade inicial do agente a respeito de um conjunto de ações associadas aos conflitos.

Para Allen, um agente mantém três descrições parciais do mundo associado ao domínio do planejamento. A primeira descrição é chamada de mundo esperado, que vem a ser o mundo que um agente espera encontrar dado um certo conhecimento a respeito de eventos futuros, ou seja, é o mundo que comprende as situações imediatas decorrentes do processo natural de transformação.

O segundo mundo é chamado de mundo desejado, e contém as descrições de um mundo segundo a vontade ou desejo do agente, que determina a idealização dos objetivos.

Finalmente, o terceiro mundo , que é chamado de mundo planejado, que vem a ser o mundo esperado sob o efeito do planejamento realizado, na tentativa de se alcançar o mundo desejado. Portanto, o sucesso do planejamento será possível,se o mundo planejado se adaptar, na medida do possível ao mundo esperado, aproximando-se do mundo desejado. Esta adaptação envolve a realização de ações que são acreditadas ou não pelo

agente, responsável pelo planejamento, ou seja :

OCCUR(plano,T) <=> (∀ ação, t. EXEC(ação,t,plano) =>
OCCUR(ação,t)) E (∀ ação, t. NÃO.EXEC(ação,t,plano) => (138)
(∀ t'.IN(t',t) => NÃO(OCUR(ação,t)))) .

O predicado EXEC(ação,plano,t) especifica que a ação deve ser incluída no plano para ser executada no tempo "t", somente se o agente acredita que as proposições que determinam a realização da ação, sejam verdadeiras no momento adequado.

- Estabelecimento de Pré-Condições:

L. Morgenstern [1988] discute em seu trabalho o relacionamento entre verdades e planos , propondo soluções para os problemas de estabelecimento de pré-condições para a execução de ações e planos em ambientes complexos . Mais especificamente , Morgenstern trata das seguintes questões:

1 - Como um agente pode raciocinar a respeito de como executar uma ação ? e

2 - Se um agente deve especificar um plano na presença de incerteza, podemos assegurar que ele, realmente, executará com sucesso este plano ?

Para resolver o problema relacionado ao conhecimento pré-condicional, necessário para se executar uma ação , Morgenstern considera algumas observações a respeito da existência de um conhecimento mínimo em agentes, suficiente a solução do problema , na seguinte forma:

. agentes precisam de ter um conhecimento procedural explícito a respeito de como conduzir as ações;

. agentes sabem como executar algumas ações básicas;

. agentes reconhecem os parâmetros associados a descrição de uma ação;

. agentes reconhecem diferentes formas de uma mesma ação e , finalmente,

. agentes sabem como compor ações diferentes, utilizando sequenciamento, condições, etc.

Para resolver o segundo problema, a respeito do sucesso na execução de planos, Morgenstern analisa, também, a questão de credibilidade das ações. Ou seja, um agente pode executar um plano sómente se ele pode ter alguma certeza sobre as ações ou eventos que terão ocorrência no planejamento. Mais precisamente, um agente executará um plano se o mesmo tiver certeza sobre as ações sobre as quais o mesmo é responsável, e puder prever, com antecedência, a ocorrência de eventos externos , assegurando a credibilidade das ações.

- Existência de um Relógio Interno:

Outra observação interessante a respeito do estabelecimento de pré-condições para a execução de ações e realização de planos, foi levantada por R.C. Moore [1980] a respeito da noção de posicionamento no tempo, por parte de um agente responsável pela execução de uma ação.

Suponha, por exemplo, que em um intervalo "t" de tempo que caracteriza determinada situação "s" uma agente esteja planejando alcançar a situação "s'". Para tanto, o mesmo deve interpretar ou reconhecer a situação atual "s" e atuar de modo a provocar a transição s -> s', dentro do período que caracteriza o intervalo "t".

A. Haas [1985], propõe uma solução para este problema através da utilização, por parte do agente, de um relógio interno em sintonia com o sistema aplicativo. Este relógio interno deve possuir alguma função discreta de tempo, de modo que , a cada leitura, a mesma seja incrementada segundo um intervalo constante.

Portanto, antes de executar a ação, que determina a transição de estados, o agente realiza a leitura do relógio, cujo valor indexará as informações temporais da base de conhecimento, permitindo que a ação correta seja executada no tempo correto. Neste sentido, podemos afirmar que cada leitura do relógio, estabelece um intervalo ou período de tempo no qual determinada situação permanece constante. Desta forma, quanto mais dinâmico for o sistema aplicativo mais frequente serão as leituras do relógio e menor será a amplitude dos intervalos de tempo.

A corretude da ação é alcançada pelo fato de que o agente , a cada leitura do relógio, revisa todas as suposições e verdades indexadas

da base de conhecimento. Ademais, se, por exemplo, a transição de estados implica na solução de um conflito, a existência do relógio permite ao agente a noção exata de tempo necessária a prévia execução da ação no tempo correto.

7.5.4 - Representação de Ações no Sistema SIGT

- Forma de Representação:

No nosso sistema, a representação de ações é considerada , inicialmente, como uma lista sequencial de procedimentos que determinam a solução de conflitos segundo a dependência direta observada ao longo do processo de elaboração do plano, e a visualização de seus efeitos pode ser observada pelo uso de uma grade extraída da tabela de tempos associada ao estado final do problema. Estes procedimentos contém um conjunto de parâmetros que permitem descrever as ações na forma de funções com argumentos, ou na forma textual. Seja, por exemplo , a lista de ações sequencias, apresentada na figura 73, relacionada ao planejamento traduzido pela grade de trens, da figura 23, seção 2.3.4.

```
 7  1  6  2 4.70E+000
 7  4  5  2 5.56E+000
 1  9  6  2 5.60E+000
 5  7  5  1 6.13E+000
 5  4  4  1 6.06E+000
 7  5  4  1 6.45E+000
 1 11  7  2 6.46E+000
 7  2  4  1 6.83E+000
 1  8  8  2 7.10E+000
 9  5  5  1 7.22E+000
 7  3  4  1 7.29E+000
 9  2  5  1 7.58E+000
 1 10  9  2 7.96E+000
11  5  6  1 7.97E+000
 9  3  5  2 8.04E+000
11  2  6  1 8.33E+000
 6  7  4  1 8.36E+000
 1 12 10  1 8.39E+000
12 10 10  1 8.41E+000
 1 12 10  1 8.54E+000
 8  5  7  1 8.72E+000
 9  6  4  2 9.04E+000
 8  2  7  1 9.08E+000
 4 12  9  2 9.43E+000
10  5  8  1 9.47E+000
10  2  8  1 9.83E+000
12  5  9  1 1.02E+001
 8  3  6  1 1.03E+001
12  2  9  2 1.06E+001
10  3  7  1 1.10E+001
12  3  8  1 1.16E+001
```

figura 73: lista de ações.

Desta forma, a partir desta lista de ações , o sistema propõe , iterativamente, ao despachador, ações planejadas, na forma da figura 74.

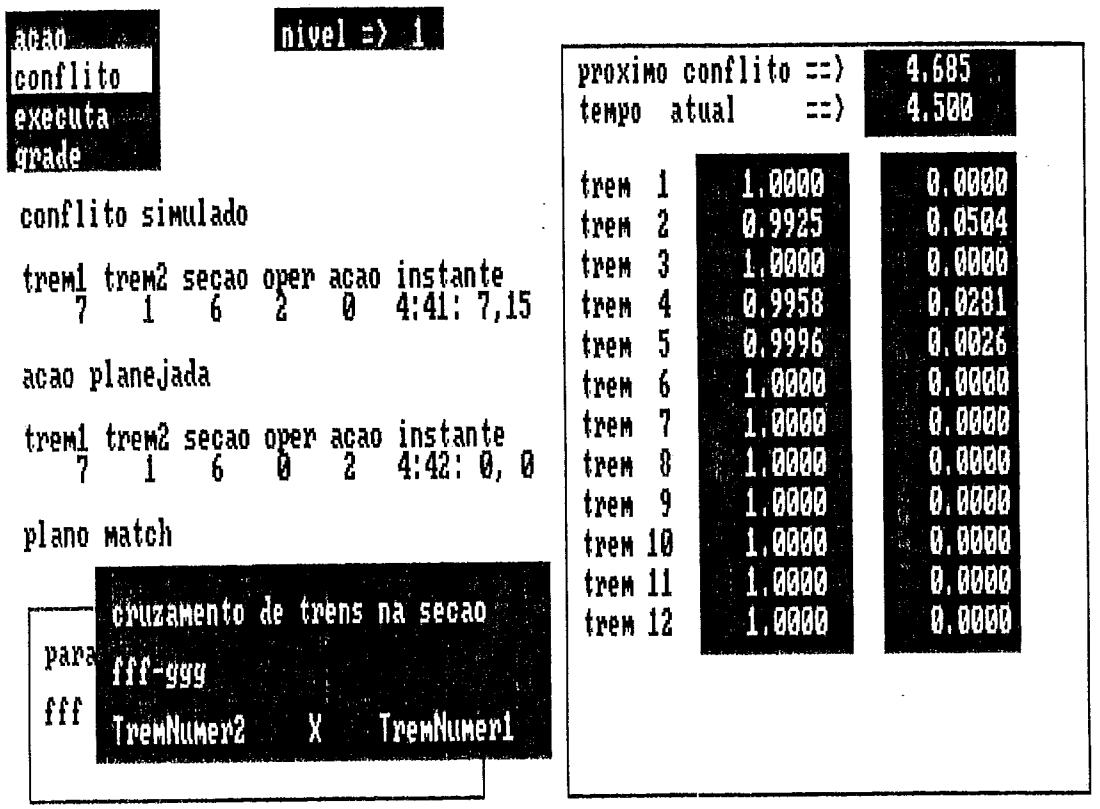


figura 74: ação planejada e conflito simulado.

No que tange a ação planejada, podemos verificar que os parâmetros trem1, trem2 e seção, caracterizados pelos seus respectivos códigos , identificam os objetos envolvidos na situação de conflito, a acontecer no instante projetado. A partir dos mesmos, é possível acessar as informações dos ojetos no sentido de descrever a ação com maiores detalhes. O parâmetro ação, determina o tipo de medida a ser tomada, a qual pode ser colocada na forma textual, facilitando a interpretação da ação por parte do controlador.

Quanto ao conflito esperado, ou conflito simulado, podemos verificar tambem, a existência dos parâmetros que especificam os objetos envolvidos no conflito, e, também, um parâmetro associado a inferência realizada pelo sistema de produção. É possível, também, descrevermos o conflito na forma textual, através do uso de janelas.

Para que ocorra o "match" ou a ligação do plano, e consequente execução da ação por parte do controlador , e necessário que :

. os parâmetros seção, trem1 e trem2 sejam, necessariamente, os mesmos na ação planejada e no conflito esperado;

. o parâmetro de inferência ou operador seja compatível com a ação proposta.

- Análise das Pré-Condições:

Neste ponto, é interessante observarmos que o sistema é capaz de prever, com a antecedência devida , a viabilidade da ação, considerando que o instante de ocorrência do conflito simulado ou esperado é calculado através de um esquema de projeção, utilizando informações que justifiquem a ocorrência do mesmo. É possível, também, assegurar ou não a verdade da ação, considerando que a inferência é realizada com base em fatos atualizados. Sendo assim, consideramos que a rotina de execução do sistema SIGT atende as pré-condições estabelecidas por Morgenstern. Devemos observar, também, que os parâmetros, relativos ao instante de ocorrência dos conflitos planejados e simulados não precisam , necessáriamente, serem iguais, mas apenas compatíveis com a ocorrência do conflito na seção específica.

Quanto as pré-condições para o estabelecimento de ações , podemos verificar que , inicialmente, o agente ou sistema de planejamento acredita na viabilidade das ações, e possui o conhecimento necessário de modo a suportar esta afirmativa. Mais ainda, possui conhecimento a respeito dos parâmetros envolvidos na ação , sabe reconhecer a mesma sob várias formas, e, também, resolver certas situações de impasse, como por exemplo, no caso da ocorrência de falhas , solicitar a rotina de execução as providências em relação ao reparo do plano.

Quanto a composição de ações, fica subtendido, para o agente responsável pela elaboração do plano, dada a hierarquia do planejamento, que a resolução de um conflito , por exemplo: conflito de cruzamento em linha singela, determina a execução de pelo menos cinco passos ou ações parcialmente ordenados, segundo os vários exemplos já apresentados em nosso trabalho. Análogamente, durante a execução do plano é de se esperar que o agente responsável, neste caso o controlador , tenha capacidade suficiente de estabelecer a composição de ações , a partir da sugestão feita pelo sistema de planejamento.

Neste ponto, queremos ressaltar que, ao contrário dos sistemas de robótica, no sistema SIGT o agente responsável pela execução de ações físicas ou comandos é o proprio controlador dado a proposta de controle e gerenciamento em laço aberto sugerida na seção 2.2.3, em favor da segurança do sistema. Neste caso, as ações do tipo avanço são permitidas ou não, através da abertura de chaves automáticas a partir de centros remotos, por parte dos controladores.

- Considerações sobre a Situação Real de Operação:

Finalmente, queremos observar que em nosso sistema, conforme a figura 74, existe um relógio interno que simula o tempo real de forma determinística e discreta, possuindo um intervalo de leitura estipulado pelo sistema. No modelo de monitoração, é possível detectar, préviamente, a ocorrência de um conflito, através de um método de projeção, e , portanto, atuar de forma correta de modo a provocar a transição de estados ou a solução de conflitos segundo a ação planejada.

Entretanto, a utilização desta rotina de execução em sistemas reais de aplicação , traria alguns problemas, quanto ao estabelecimento do momento ideal de execução das ações.

Sabemos que, para a confiabilidade e precisão da informação, quanto menor for o intervalo de leitura do relógio, ou quanto mais perto estivermos do mundo esperado, mais confiáveis serão as suposições ou verdades a respeito do estado atual, bem como, maior certeza a respeito da não ocorrência de eventos externos indesejáveis. Por outro lado, caso fosse necessário a realização de replanejamento ou reparo do plano com base na situação presente , teríamos um período de tempo de curta duração.

Além do mais, dada as características do sistema aplicativo, para cada situação de conflito , existe um tempo de segurança que antecede ao tempo de ocorrência do mesmo (determinando o momento ideal de avaliação da ação e execução do comando) que deve ser respeitado. Ou seja, se determinada ação provoca o desvio de um trem em movimento para um pátio de cruzamento, conforme a situação descrita na figura 75, o controlador (agente) deve fazer a abertura de chave (execução do comando) antes que o trem passe pelo sinal automático que precede o

sinal controlado (vermelho) associado a posição da chave. Desta forma, ao passar pelo sinal automático, o trem recebe o sinal amarelo, e inicia o processo de redução da velocidade, para que o desvio para o pátio seja feito com maior segurança.

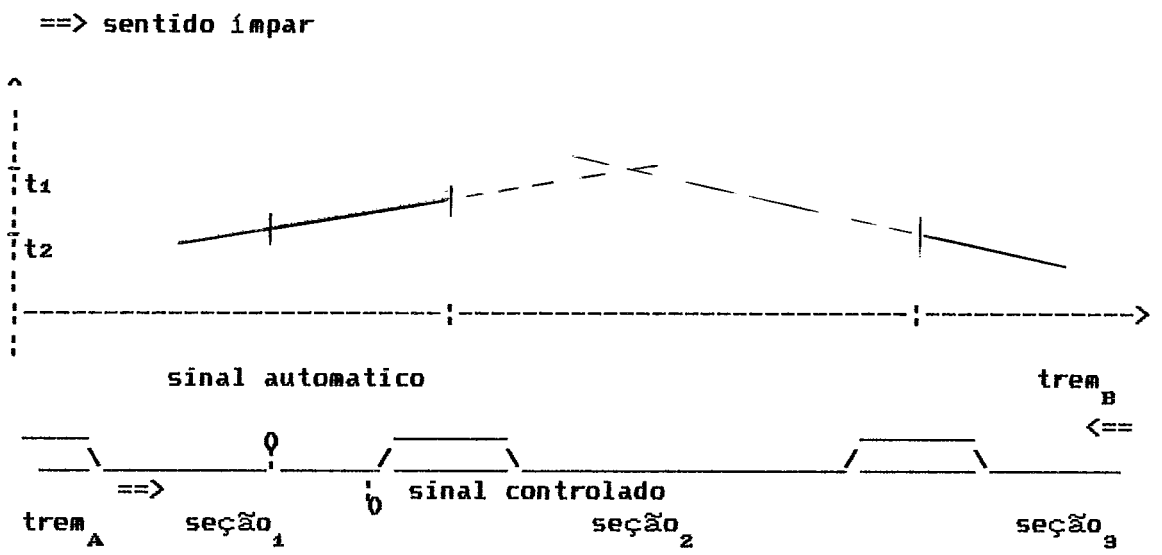


figura 75: tomada de decisão em situação de conflito.

Podemos observar, que este intervalo deveria ser igual a diferença dos tempos t_1 e t_2 .

Entretanto, dada a presença simultânea de diversos trens e de diversas situações de conflito, seria impossível determinarmos um intervalo de leitura único e preciso para todo o sistema, que não comprometesse a realização de replanejamento.

A solução proposta em nosso sistema, consiste em desassociar a avaliação de ações planejadas e a execução de comandos, da leitura do relógio. Neste sentido, tanto no modelo de monitoração simulado, quanto no modelo real, é proposto um intervalo de leitura ou intervalo de monitoração cujo objetivo maior é possibilitar o controle de parâmetros do sistema, como exemplo, o nível de satisfação dos objetivos.

A atualização de informações da base de conhecimento, se dará segundo a realização de cada ação por parte do agente e também,segundo o processo de orientação baseado na ocorrência de eventos externos,através da execução de interrupções com prioridade. Desta forma, teremos sempre um retrato fiel das transformações ocorridas no sistema, possibilitando

a validação da base de conhecimento durante a avaliação das ações.

Neste sentido, será sempre oferecido ao controlador, pela rotina de execução do sistema , um conjunto de ações planejadas segundo a ordem de urgência e prioridade. Este conjunto de ações será alterado , iterativamente , sempre que houver a execução de um comando , ou a proposta de modificações, resultante de medidas de reparo ou da realização do replanejamento.

7.5.5 - Gerenciamento de Restrições Temporais

Últimamente, tem se desenvolvido um grande número de sistemas cujo objetivo maior é permitir a manutenção de base de dados temporais , a realização de inferências nestas bases de dados, e também, a utilização de algoritmos baseados em propagação de restrições como forma de atualizar a base de dados face a introdução de novas restrições temporais, ou seja , restrições que , de alguma forma, restrigem ou estabelecem um relacionamento na ordem de ocorrência de eventos.

De um modo geral , a base de dados destes sistemas é baseada na utilização de grafos ou redes associativas segundo as descrições feitas na seção 7.5.1. Nesta linha de sistemas, destacamos os trabalhos de S.A. Vere [1983], C.E. Bell [1989], T. Dean [1988] e T. Dean, D. McDermott [1987]. Existem também, alguns sistemas que fazem uso de linguagens do cálculo de predicados, como no trabalho de A. Montfroglio [1988], que desenvolveu um sistema PROLOG voltado para a solução do problema de planejamento de horários com restrições temporais e restrições de recurso.

No entanto, daremos ênfase nesta seção, aos trabalhos de Vere[1983] e Bell[1989], que apresentam o desenvolvimento de sistemas de planejamento semelhantes ao SIGT. Estes sistemas consideram, para a representação de dados, redes do tipo PERT , bem como a existência de conflitos associados a utilização de recursos e a satisfação de restrições temporais que estabelecem precedências na ordem de execução de atividades.

- Sistemas com Restrições na Forma Disjuntiva:

Colin E. Bell[1989] apresenta um sistema de planejamento que representa planos como uma rede parcialmente ordenada, possuindo como vértices instantes de tempo, e como arcos restrições de precedência, na forma de uma rede PERT.

O processo de planejamento, voltado para a solução de problemas de "scheduling" com restrições de recurso , consiste de um processo hierárquico com passos de refinamento tendo como plano esqueleto uma rede PERT inicial.

Desta forma, iterativamente, o sistema detecta no plano parcial um conflito em relação a uma restrição de recurso, e propõe a adição de restrições na forma disjuntiva, como forma de resolver o conflito, tornando o plano cada vez mais restrito.

A rede PERT inicial, é formada determinando-se o tempo crítico programado para um projeto, em função do tempo de duração e das restrições de precedência das atividades envolvidas. Portanto, considerando a existência de um conjunto de recursos a serem compartilhados por várias atividades, o sistema, iterativamente , acrescenta uma restrição disjuntiva como forma de quebrar o conflito existente, e, em seguida, determina um novo plano imcompleto e viável para o conjunto de restrições existentes. Na certa, quando não houver mais a existência de conflitos associados as restrições de recurso , o plano produzido será completo e totalmente viável .

A maior crítica que podemos estabelecer ao trabalho de C.E. Bell, e a muitos outros sistemas de planejamento que operam desta forma, é que os mesmos não possuem capacidade de elaborarem planos ótimos. Isto se atribui a inexistência de uma estrutura que permita estabelecer um domínio de planos parcialmente completos, a ausência de relações de preferência e, também, a inexistência de uma prova de dominância explícita.

Neste sentido, Bell, comparando seu sistema a sistemas que permitem a obtenção de planos ótimos , como o sistema de J.P. Stinson, E.W. Davis e B.M. Khumawala[1978] , argumenta que os dois tipos de planejamento são complementares, no sentido de que , seu sistema trabalha mais, eficientemente, com poucas restrições de recurso, ou

conflitos, enquanto que os sistemas que se baseiam em técnicas de busca trabalham mais, eficientemente, com mais restrições.

Infelizmente, discordamos deste ponto de vista, ao que parece, contrário a filosofia de solução de problemas na visão da Inteligência Artificial, a qual representa uma extensão a filosofia tradicional de solução de problemas segundo os modelos de Pesquisa Operacional.

- Sistema de Planejamento DEVISER:

. representação de atividades:

O sistema DEVISER, desenvolvido por S.A. Vere[1983] , consiste de um programa de computador voltado para a geração de plano de atividades , a exemplo de uma rede PERT, com aplicações no planejamento e programação de atividades espaciais incluindo fotografia e transmissão de informações.

O sistema considera uma atividade, associada a uma determinada ação ou passo, caracterizada por um período fixo de duração, possuindo, a exemplo do sistema STRIPS , a forma de uma regra de transformação . Cada atividade contém, no lado esquerdo, uma lista de literais que não são modificados pela ação e uma lista de antecedentes ou pré-condições que são deletadas; e, no lado direito, uma lista de consequentes cujas proposições são adicionadas a base de fatos assim que a mesma se processa.

Os objetivos do problema são representados na forma de uma conjunção de proposições contendo, entretanto, restrições temporais na forma de janelas de tempo. Estas janelas possuem três parâmetros, considerando o tempo mais cedo, o tempo ideal e o tempo mais tarde no qual o objetivo deva ser alcançado. A situação inicial é representada na forma de uma lista de proposições verdadeiras associada a um instante de tempo.

No sistema DEVISER, e possível a programação de eventos determinísticos associados a um tempo específico de ocorrência. Estes eventos , são, também , tipos de atividade , que podem adicionar ou deletar fatos ao domínio do problema no momento especificado.

Entretanto, a lista de antecedentes deste tipo de atividade é sempre vazia, já que não é possível a representação de eventos condicionais.

Um terceiro tipo de atividade se refere a realização de inferências, que consistem em tipos de ações nas quais a verdade dos fatos acrescidos ao domínio, depende da continuidade da verdade das premissas ou pré-condições, exigindo, portanto, um sistema de manutenção de verdade.

. técnica de planejamento:

Básicamente, o sistema trabalha segundo o processo regressivo de construção de planos a partir dos objetivos do problema, gerando uma rede de atividades da seguinte forma:

Inicialmente é construída uma rede inicial contendo a situação inicial , os objetivos , os eventos programados e a situação final. Em seguida, executasse:

1 - Criação de um arco:

Se alguma condição de um objetivo é atendida por qualquer nó (eventos, ação, objetivo, inferência ou situação inicial) existente na rede , então é criado um arco ligando o nó ao objetivo;

2 - Expansão de um nó:

Se um sub-objetivo não pode ser satisfeito pelo primeiro passo, identifica-se uma atividade (ação) que o alcance, através da geração de um nó contendo, como proposições, a negação dos literais da lista de antecedentes e os literais da lista de consequentes . Óbviamente, as pré-condições da ação se tornam novos sub-objetivos;

3 - Deteção de conflito, e solução por ordenamento de atividades paralelas:

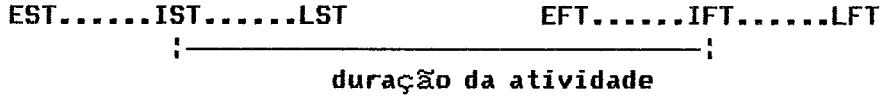
A exemplo do sistema NOAH , o sistema DEVISER detecta a presença de conflitos (por exemplo, a existência de proposições contraditórias em relação a um mesmo literal em atividades paralelas) através de críticas e propõe a sua solução através do ordenamento de atividades.

. raciocínio temporal:

Ao que nos parece, a grande novidade do sistema DEVISER se refere a sua capacidade de raciocínio temporal, considerando a possibilidade de se especificar intervalos de tempo ou janelas que restrigem o início e fim de atividades, o alcance dos objetivos e a programação de eventos. Para tanto, o sistema realiza uma série de procedimentos no sentido de propagar as restrições temporais ao longo da rede, durante o processo de construção de planos.

Em nosso trabalho, descreveremos apenas o procedimento associado a compressão de janelas , provocadas pelo ordenamento parcial de duas atividades. Os procedimentos restantes, referentes a propagação da compressão de janelas, poderão ser visto no trabalho de Vere.

Primeiramente, seguindo o trabalho de Vere, vamos considerar a existência de janelas, associadas a uma atividade, com a representação descrita na figura 76.



onde :

EST se refere ao tempo de início mais cedo;
IST ao tempo ideal de início;
LST ao tempo de início mais tarde;
EFT ao tempo de término mais cedo;
IFT ao tempo de término ideal, e
LFT ao tempo de término mais tarde.

figura 76: representação de janelas associada a atividade.

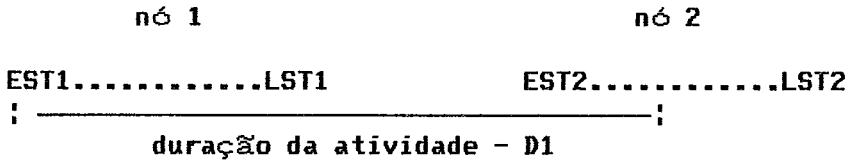


figura 77: nós não sequenciais.

Desta forma, se dois nós não ordenados, ou não sequenciais, ver

figura 77, forem ordenados sequencialmente , deve-se restabelecer os tempos de início da atividade, através dos seguintes procedimentos:

- contração do tempo de início mais tarde do primeiro nó:

```
LFT1 <- LST1 + D1;
Se LFT1 > LST2 Então
   LST1 <- LST2 - D1;
```

- contração do tempo de início mais cedo do segundo nó :

```
EFT1 <- EST1 + D1;
Se EFT1 > EST2 Então
   EST2 <- EFT1;
```

Portanto, para nós sequênciais, teremos:

$$EST1 + D1 \quad EST2 \text{ e} \qquad (139)$$
$$LST1 + D1 \quad LST2 \text{ ,}$$

e, para nós consecutivos :

$$EST1 + D1 = EST2 \text{ e} \qquad (139)$$
$$LST1 + D1 = LST2 \text{ .}$$

Após os procedimentos acima, determina-se os tempos finais, considerando a existência dos tempos ideais, ou seja:

```
Se IST1 < EST1 Então
   IST1 <- EST1
SenãoSe  IST1 > LST1 Então
   IST1 <- LST1;
LST1 <- IST1;
EST1 <- IST1;
```

Para nós sequenciais, a determinação dos tempos finais de início , não afeta os tempos do segundo nó. Entretanto, para nós consecutivos, teremos, que considerar:

$$EST2 = LST2 = IST1 + D1. \qquad (140)$$

Finalmente, queremos destacar que na existência de duas janelas, a serem conciliadas durante a expansão de um nó, deve-se fazer:

$$EST = Max \{ EST1, EST2 \} \text{ e} \qquad (141)$$
$$LST = Min \{ LST1, LST2 \} .$$

O procedimento acima, consiste, justamente, no procedimento de determinação dos tempos de início mais cedo e mais tarde das atividades

de uma rede PERT. Portanto, o sistema DEVISER pode ser considerado como um sistema de planejamento que permite a realização de simulações determinísticas, através da programação de eventos, orientados segundo o alcance de objetivos com restrições temporais. O produto final é, como vimos, uma rede PERT com um conjunto de atividades parcialmente ordenadas que determinam um caminho crítico associado a um conjunto de atividades paralelas.

Quanto ao sistema SIGT, queremos lembrar que o mesmo, a exemplo do sistema DEVISER , permite a associação de restrições temporais aos objetivos do problema. Também, como tratamos de um problema de minimização, mantemos sempre a representação do tempo de início das atividades associadas aos percursos dos trens, na tabela de tempos, estabelecendo um conjunto de restrições ativas para o problema de sequenciamento de trens. A tabela final, como sabemos, representa um conjunto de atividades , parcialmente ordenadas , formando um conjunto de caminhos críticos, que atendem ao percurso de cada trem.

7.6 - Execução de Planos

Considerando a classificação do sistema SIGT como um sistema de monitoração e replanejamento como forma de reagir a imprevisibilidade do sistema aplicativo, durante o processo de execução de planos, vamos, nesta seção, discutir alguns aspectos relacionados as principais técnicas utilizadas no sentido de possibilitar ou dar continuidade a execução do planejamento.

7.6.1 - Considerações

Segundo D. McDermott [1988], a distinção entre planejamento e execução é devido ao fato de que o processo de elaboração de planos, sempre antecede o processo de tomada de decisões relativo a execução das ações planejadas.

Entretanto, muitos pesquisadores, D. Chapman, P. Agre , M.J. Schoppers entre outros , tem abandonado este pensamento, no planejamento de sistemas aplicativos e altamente dinâmicos. Neste caso, tem-se

destacado o desenvolvimento de estratégias de planejamento reativo, ou seja, o desenvolvimento de métodos que permitam ao agente empreender uma conduta competitiva, face a imprevisibilidade do domínio, envolvendo a realização simultânea de planejamento e execução.

Contudo, é consensualmente aceito, em certos domínios aplicativos, a existência do processo de planejamento na forma tradicional; e, neste caso, para que a execução se torne viável, é necessário estabelecermos:

1 - As pré-condições que permitam a execução de planos e ações por parte do agente, já analisadas, anteriormente, na seção 7.5.3;

2 - Se o sistema aplicativo ou domínio de planejamento é favorável ao desenvolvimento de planos, e

3 - Qual o processo ou técnica que permitirá ao sistema de planejamento , ou agente, lidar com a incerteza e dinâmica das situações reais do domínio, durante a execução dos planos.

- Domínio Favorável ao Planejamento:

Considerando que a primeira questão, já foi devidamente discutida em seções anteriores de nosso trabalho, vamos, inicialmente, discutir sobre o domínio ideal de planejamento.

É evidente, que os sistemas de alta previsibilidade, são os mais favoráveis a realização de planejamento, ver seção 7.4.1 . Dada a inexistência de incerteza, é possível, nestes sistemas, raciocinar sobre futuros eventos e sobre o efeito das ações , com base no cálculo de situações (J.McCarthy e P.J. Hayes[1969]).

Entretanto, como já dissemos, são poucos os sistemas aplicativos que se enquadram nesta categoria. Portanto, devemos ser mais realistas, e refazermos nossa opinião, a respeito das propriedades desejáveis de domínios para a realização de planejamento.

Segundo comentários de R. Dreyfus [1989], em um painel sobre o futuro das aplicações de sistemas de Inteligência Artificial, os programas de IA tem grande adaptação em problemas onde o agente se organiza em termos de planos e objetivos. Como exemplo, Dreyfus destaca

os domínios estruturados altamente combinatórios , caracterizados pelo fato de haver um certo conhecimento sobre o efeito das ações , que são geradas de forma racional, ou seja, dotadas de objetividade.

Dreyfus considera, também, nestes domínios, a existência de um número elevado de situações ou estados, onde a intuição desenvolvida a partir de experiências , não contribue, significativamente, para o desenvolvimento de excelentes especialistas. Neste caso , é possível que programas desenvolvidos com base em regras ou unidades de conhecimento, associados a utilização de um maior esforço de busca, tenham um desempenho satisfatório.

Para Dreyfus, a intuição vem a ser a capacidade de associar, de forma rapida e sem esforço, situações reais a ações ou decisões com certa objetividade, adquirida a partir de experiências anteriores e bem sucedidas. Ou seja, neste nível de raciocínio, o agente não só reconhece a situação , como, também, o modo de agir corretamente , tomando como base certas experiências.

Segundo T. Winograd [1989] esta capacidade de raciocínio é altamente desejável em sistemas que deparam, sempre, com situações inesperadas, exigindo modelos de planejamento altamente reativos.

Desta forma , podemos considerar os sistemas que desenvolvem projetos, ou configurações, como exemplo o sistema XCON , como sistemas altamente desejáveis ao uso de programas de IA e técnicas tradicionais de planejamento. Por outro lado, o jogo de xadrez, embora seja um domínio estruturado e altamente combinatório, falha no que se refere ao aspecto intuitivo do raciocínio.

Como exemplo, podemos citar o desenvolvimento do programa "Deep Thought" na Universidade de Carnegie-Mellon. Apesar da alta capacidade de análise, que permite ao programa analisar aproximadamente 800000 posições de jogo (lances) por segundo de partida, o mesmo foi fácilmente vencido, recentemente, em outubro de 1989, pelo campeão mundial de xadrez, o soviético Gary Kasparov. Esta derrota, creditada por muitos, pela ocorrência de vários erros de estratégia, interrompeu uma série de 500 partidas invictas, contra vários jogadores, incluindo alguns, da categoria de mestres mundiais.

Entretanto, se considerarmos os sistemas de planejamento e "scheduling", como o sistema SIGT, vamos verificar, que, face a complexidade combinatória do problema, o uso da intuição pouco contribui para o aumento do desempenho quanto ao alcance dos objetivos.

No sistema de controle e gerenciamento de trens, podemos considerar que a capacidade intuitiva, adquirida de experiências, contribui, de certa forma, para o aumento de segurança das medidas de controle. Entretanto, pouco contribui para a eficiência em termos de otimalidade. Isto se deve, principalmente, ao fato de que , ao atuar de forma reativa, segundo uma conduta que reage a situações, o controlador não desenvolve linhas de raciocínio. Também, as situações de conflito, em si, não são tão complexas a ponto de justificar este tipo de comportamento.

Portanto, consideramos que, no problema de gerenciamento do tráfego de trens, o caráter enumerativo ou a complexidade combinatória, excede a complexidade de situações específicas, justificando a utilização de sistemas de planejamento, se devidamente observada a segurança do movimento.

Desta forma, se o sistema for competente para lidar com os aspectos adverso do sistema aplicativo , como exemplo: incerteza, situações inesperadas, etc., teremos uma grande chance de realizarmos, com , com sucesso, a automação parcial do servico de despacho de trens.

D. McDermott [1989], destaca, também, como essencial ao sucesso do planejamento, a possibilidade de simularmos realísticamente, um domínio semelhante, considerando as características e níveis de detalhe principais do sistema aplicativo.

- Técnicas de Execução:

Analisando agora , a última questão, que trata da existência de técnicas de execução, faremos algumas observações considerando o trabalho de D. McDermott[1989].

Como já foi dito, a existência de um tempo de planejamento e de um tempo de execução se deve ao exercício da reflexão, que define dois períodos distintos, sendo que, o período de planejamento deve anteceder

o período de execução. Ou seja, a reflexão sobre a forma de atuar ou o raciocínio sobre ações, antecede o tempo de execução, assim como o exercício do meta-planejamento antecede o exercício do planejamento.

Entretanto, na ocorrência de falha no processo de execução, que atitude devemos tomar para que a realização dos planos seja levada adiante ?

A primeira vista, podemos pensar em uma técnica semelhante ao processo de revisão ou transformação de planos, ou seja, dado uma situação inicial indesejável, que vem a ser a situação de falha, elaboramos um conjunto de ações que permitam dar continuidade a execução do planejamento.

Este processo, é conhecido como replanejamento, podendo implicar na total reelaboração de um novo plano, incluindo revisão nas prioridades, objetivos, pre-condições, etc., ou, sómente, no reparo de falhas, considerando, por exemplo, a simples adição de passos ou ações corretivas. Na verdade, a elaboração de um novo plano ou a realização de uma simples medida de reparo, dependerá do tipo de falha e do tempo disponível a realização de tal processo.

Portanto, a princípio, podemos considerar o planejamento, ou o replanejamento, como coisas iguais. De fato, em sistemas dinâmicos e em tempo real, os dois processos se confundem , tendo como única diferença, a situação, a partir da qual, o processo de planejamento é reiniciado, e o tempo disponível para a realização do mesmo, face a existência de rígidos tempos de resposta.

Na realização do replanejamento, a situação inicial vem a ser a ação que caracteriza a existência de falha (a ocorrência ou não dá falha dependerá da dinâmica do sistema aplicativo), e o tempo disponível fica restrito pela necessidade do novo planejamento. Por outro lado, o planejamento contínuo, sem replanejamento, pode ser considerado como a integração de planos bem sucedidos, consecutivos no tempo, tendo, neste caso, maior flexibilidade quanto ao tempo necessário a sua reelaboração.

Considerando que a existência de situações inesperadas produz o surgimento de novas informações, e face a dinâmica do sistema

aplicativo, surge a necessidade de constantes revisões de planos, na medida em que são propostos passos de restabelecimento. Desta forma, alguns sistemas de planejamento fazem uso das informações mais recentes e propõem o desenvolvimento de planos de forma oportunista, intercalado com as fases de execução. Esta forma de execução, consiste de uma forma conjunta de monitoração e replanejamento em tempo real, tendo influenciado bastante, a técnica de execução desenvolvida para o sistema SIGT.

Contudo, na ausência do planejamento tradicional, ou na visão de planejamento como o exercício de uma conduta competente, devemos observar que não se verifica a intercalação sucessiva entre intervalos de planejamento (elaboração de planos) e intervalos de execução, já que o agente está sempre atento, ou reagindo a ocorrência de situações. Portanto, se esta atuação ou conduta é competente, não haverá a existência de falhas, e , consequentemente, a necessidade de replanejamento.

Em domínios de planejamento onde a ocorrência de situações inesperadas possui certa previsibilidade e, considerando què o agente seja capaz de detectar a ocorrência de falhas, é natural que o sistema de planejamento desenvolva uma forma eficiente de reagir a estas situações, de modo a garantir o andamento da execução, se possível, dentro da mesma linha de planejamento, levando a uma nova técnica de execução.

Como exemplo deste domínio, podemos destacar aqueles cujo agente se orienta pela solução contínua de conflitos. Neste caso, a ação remediadora por parte do agente deve anteceder a ocorrência dos conflitos, evitando deste modo, situações altamente desastrosas em alguns sistemas aplicativos.

Entre as formas mais comuns de um sistema reagir a situações de falha na presença de planejamento, destacamos a realização de medidas de reparo pré-estabelecidas, ou seja, através da introdução de passos que restaurem a viabilidade do planejamento, dentro da mesma linha de raciocínio. Devemos destacar, também , a realização de planos contigentes, que constituem alternativas de planejamento "ad hoc" em situações de falha, possuindo linhas de raciocínio próprias; e , finalmente, a realização de planos condicionais, que permitem ao sistema

reagir a situações de falha, observando, em tempo de execução , a existência de certas condições.

Finalmente, queríamos lembrar que existe um certo consenso na área de planejamento, de que algum critério de otimalidade para o exercício do replanejamento em tempo de execução, deve levar em conta o tempo disponível e a rigidez do tempo de resposta. Sendo assim, queremos ressaltar a necessidade e importância de se estabelecer o período ideal de planejamento segundo a imprevisibilidade do sistema aplicativo, que dita a frequência entre falhas e conflitos e, consequentemente , o tempo disponível em função da necessidade de respostas.

É necessário considerarmos, também, que enquanto se processa o replanejamento, dada a dinâmica do sistema aplicativo, surgem novas informações, que podem, de alguma forma, comprometer a reelaboração de planos. Para tanto, segundo McDermott, é fundamental que o tempo necessário a realização do replanejamento seja inferior ao tempo disponível, ou seja, inferior aos intervalos de transição entre estados.

No sistema SIGT, este intervalo é caracterizado pelo intervalo médio entre conflitos, que dependerá, no entanto, da frequência de trens na malha. Se o tráfego de trens estiver altamente saturado, teremos para uma taxa de 4 trens/hora em média, um conflito a cada 15 min., o que representa um tempo muito superior ao tempo necessário a execução do replanejamento, que gira em torno de 1 min.

- Monitoração:

Neste tópico, queremos ressaltar o papel da monitoração, ou seja, o acompanhamento da execução dos planos, sobretudo, quanto ao efeito das ações e a existência de pré-condições, na realização ou execução do planejamento.

A medida em que monitoramos um plano, através da avaliação de parâmetros, que podem ser obtidos, por exemplo, de informações sensorizadas , verificamos a execução ou andamento do planejamento. Entretanto, caso as informações disponíveis indiquem a existência de uma falha, o sistema de monitoração passa o controle a rotina de execução,

que através do uso dos parâmetros de monitoração, deteminará ou não a revisão de prioridades, pré-condições, etc. , de modo a realizar o replanejamento a partir da situação de falha.

Desta forma, e possível prosseguir o processo de planejamento em novas direções, procurando, sempre, atingir os objetivos inicialmente estabelecidos. Neste sentido, segundo D. McDermott[1988], os processos de planejamento e replanejamento se confundem, sendo delineados pela dinâmica e incerteza do sistema aplicativo, com o único objetivo de dar continuidade, de forma racional, a execução do planejamento.

7.6.2 - Planejamento Contingente (T.A. Linder e J. Glicksman [1988])

- Introdução:

No trabalho de Linder e Glicksman, já citado anteriormente, é proposto um sistema de planejamento para a elaboração de rotas para um veículo autônomo em um terreno com obstáculos, na presença de incerteza, face a possibilidade de obstrução de caminhos ou regiões.

O modelo proposto por Linder e Glicksman emprega um algoritmo de busca A* hierárquico, que determina a rota preferida baseando-se no desenvolvimento de uma função de avaliação que penaliza as rotas que tenham dependência de eventos críticos para o sucesso do plano, e computa o custo de rotas alternativas ou planos contigentes na presença de obstruções.

Desta forma, é possível ao sistema de planejamento reagir, em tempo de execução, a presença de certas situações de falha. A utilização de planos contingentes, se justifica como forma de evitar a elaboração de planos alternativos durante a fase de execução. Entretanto, como já dissemos anteriormente, este processo só será válido sob certas condições de previsibilidade.

- Função de Custo e Expansão de Planos:

A admissibilidade da heurística é baseada no fato de que, a medida em que níveis de detalhe (planos contingentes) são acrescidos em rotas candidatas, o valor da função cresce monotônicamente.

Também, para que o carater monotônico da função seja preservado, é necessário que o valor dos custos reais seja igual ou superior ao valor dos custos estimados de roteamento. Desta forma , ao se computar o custo entre duas regiões, considera-se como estimativa o custo relativo a distância euclidiana entre os dois pontos. Por outro lado , o custo real é determinado através de um algoritmo de programação dinâmica, que computa o melhor caminho baseado em informações a respeito do terreno ou do domínio de planejamento. Lógicamente, o custo estimado será sempre menor do que o custo real.

Na construção de planos contingentes, considera-se a existência de um valor de probabilidade associado a ocorrência de uma possível falha ou obstrução. Desta forma, na elaboração de planos é computado o custo probabilístico de planos contingentes. Ou seja, se determinada região A alcança uma região B, relativo a rota:

(SIT A B ... OBJ) , (142)

onde:
SIT se refere a situação inicial, e
OBJ ao objetivo,

temos como valor da função de avaliação, o custo real de se atravessar as regiões A e B, C_A e C_B, mais o custo real de se atravessar de uma região para outra, C_{AB}, mais uma estimativa de custo até o objetivo, E_{B*}.

Desta forma, se este plano imcompleto, estiver no nível máximo (sem possibilidade de planos contingentes) podemos expandí-lo, considerando todas as regiões alcançáveis de B, ou seja, são gerados os planos na forma:

(SIT A B X ... OBJ), (143)

com custo $C_A + C_B + C_X + C_{AB} + C_{BX} + E_{X*}$.

Entretanto, se fosse considerado a possibilidade da existência de uma obstrução na região B, com probabilidade "p", haveria a necessidade de se introduzir um plano contingente, no plano anterior, na forma:

(SIT A B (... OBJ) X ... OBJ) , (144)

com custo total:

$C_A + C_B + C_{AB} + (1-p).(C_X + C_{BX} + E_{X*}) + (p).(E_{B*})$.

Mas , se a região X alcança o objetivo , conforme o plano:

(SIT A B (...OBJ) X OBJ) , (145)

com custo:

$C_A + C_B + C_{AB} + (1-p).(C_X + C_{BX} + C_{X*}) + (p).(E_{B*})$

e, considerando a geração de um novo plano , na forma:

(SIT A B (Y ...OBJ) X OBJ), (146)

deve-se, recalcular o custo anterior, derivando :

$C_A + C_B + C_{AB} + (1-p).(C_X + C_{BX} + C_{X*}) + (p).(C_{BY} + C_Y + E_{Y*})$.

- Considerações:

Para nós, esta técnica de introdução de planos contingentes é semelhante ao processo de replanejamento em tempo de execução, se o mesmo fosse feito a priori. Isto porque, o algoritmo de Linden e Glicksman parte de situações de falha (regiões intransponíveis) e caminha no sentido de formular rotas alternativas , segundo uma nova direção de planejamento, de modo a alcançar o objetivo. Ou seja, não é garantido, como nas medidas de reparo, a preservação de parte do planejamento situado no primeiro nível de hierarquia.

Portanto, a única diferença desta técnica quanto a continuidade do planejamento, em relação a técnica de replanejamento tradicional, é que os planos contingentes são idealizados em tempo de planejamento,não havendo, portanto, a intercalação sucessiva entre períodos de replanejamento e períodos de execução.

Quanto a questão da otimalidade do planejamento realizado, temos uma garantia inicial da realização do melhor plano face a ocorrência de falhas que levam a utilização de planos contingentes. Ou seja, pelo processo de construção empregado, teremos em mão a melhor alternativa de planejamento. Entretanto, caso não ocorram as falhas previstas , teríamos, com certeza, planos alternativos de melhor valor, se desconsiderassemos a presença de planos contigentes.

Da mesma forma, a utilização de uma função de custo probabilística, penaliza as situações de falha que são críticas para o replanejamento, ou seja, nas quais o custo de rotas alternativas é excessivamente alto, aumentando a robustez do mesmo. No entanto, devemos considerar, também, que esta técnica de planejamento trabalha com menos informações que a técnica de replanejamento tradicional, a qual pode fazer uso das informações adicionais obtidas durante a fase de

execução.

Portanto, a utilização da técnica de planejamento e execução proposta por Linden e Glicksman, fica condicionada a diversos parâmetros e características do sistema aplicativo, como por exemplo: previsibilidade, reatividade, sensoramento, tempo de resposta, período de planejamento, etc. . Desta forma, se torna impossível fazermos ponderações mais precisas a respeito de suas vantagens e deficiências.

Entretanto, apesar de não realizarmos testes qualitativos, achamos que a técnica de planejamento e execução baseada nos processos de monitoração e replanejamento em tempo de execução, se enquadra melhor ao sistema de gerenciamento e controle de trens, dado a existência de um tempo disponível, suficiente para a elaboração de novos planos com base em informações mais recentes ou para a adoção de medidas de reparo que restaurem a viabilidade do planejamento.

7.6.3 - "Scheduling" Oportunístico (P.A. Newman e K.G. Kempf [1985])

- Introdução:

No trabalho de Newman e Kempf, também já citado anteriormente, é apresentado um sistema de planejamento voltado para a programação ou "scheduling" de atividades associadas a tarefas de robos em um centro de produção de manufaturados.

Segundo os autores, qualquer esforço na elaboração de planos a partir do uso de suposições ou de técnicas de previsão é , muitas vezes, pouco compensativo, dado ao alto custo do processo de revisão de planos durante o período de execução.

Desta forma, Newman e Kempf propõem uma técnica de planejamento que seja capaz de produzir planos de alta confiabilidade , baseada em estratégias do tipo "least commitment", ou seja, as decisões ou ações de programação são feitas sómente quando não existe mais indícios de que as mesmas serao abandonadas, o que dependerá, essencialmente , da veracidade dos dados utilizados no planejamento. Daí, a realização de

uma técnica de planejamento do tipo oportunista, considerando a elaboração, sómente, de planos realistas.

- Estratégia Básica:

A respeito da realização de planejamento, nas tarefas de programação na produção de manufaturados, quanto a utilização de informações, Newman e Kempf, destacam a existência de três estratégias básicas:

1 - Realização de planejamento na forma estratégica, baseado em dados históricos como forma de prever as condições nas quais se executarão os planos;

2 - Realização de Planejamento em tempo real, considerando a utilização de regras de decisão, como exemplo: disciplinas de atendimento na solução de conflitos, e, finalmente,

3 - Realização de um planejamento dinâmico e oportunístico , baseado nas informações mais recentes ou confiáveis.

- Emulação em Tempo Real:

Como sabemos, a primeira estratégia de planejamento se mostra inviável em sistemas dinâmicos de alta imprevisibilidade. Por outro lado, a segunda estratégia perde o senso de otimalidade devido ao caráter imediatista e localizado do processo decisório.

Desta forma, a terceira técnica de planejamento surge como uma alternativa flexível, empregada no sentido de garantir um planejamento robusto e dotado de certa otimalidade. Para tanto, é necessário que o processo de elaboração/reelaboração de planos seja feito com rapidez suficiente, atendendo as solicitações do sistema aplicativo em relação ao tempo de resposta.

Newman e Kempf consideram esta estratégia de planejamento como sendo uma emulação do sistema aplicativo mais rápida que o tempo real. Ou seja, é proposta uma técnica de elaboração de planos, em tempo real, baseada nas informações mais recentes, possibilitando, desta forma , uma estimativa bem real do mundo planejado, em um tempo compatível com a necessidade das ações de planejamento.

Na produção de manufaturados, se considerarmos o emprego de ações associados a duração de atividades, teremos que ter as ações de planejamento antes do início da próxima atividade.

Por outro lado, os autores, em seu trabalho, não deixam bem claro o período de planejamento realizado, em função da dinâmica e imprevisibilidade do domínio aplicativo. Entretanto, é possível verificarmos que o sistema de planejamento procura desenvolver linhas de raciocínio, no sentido de alcançar certos objetivos associados ao problema em questão.

Sendo assim, podemos considerar esta técnica de planejamento semelhante a técnica proposta pelo sistema SIGT, ou seja, o exercício de replanejamento em tempo real, intercalado com fases de execução. O senso oportunista é garantido pela constante atualização da base de conhecimento, em função das ações realizadas e das mudanças verificadas no sistema aplicativo. É possível, por exemplo, uma constante atualização das informações relativas a duração das atividades, permitindo ao sistema gerador de planos trabalhar com dados mais realistas.

- Modelo de Monitoração:

É importante ressaltarmos, também, a existência de um sistema de monitoração que, em tempo de execução, deve sinalizar a realização do replanejamento, e checar, a cada instante, a viabilidade das ações.

Neste sentido, a exemplo de nosso sistema, os autores, também propõem o desenvolvimento de um simulador de planos, que permite avaliar a viabilidade do sistema de planejamento. Para tanto, o simulador faz uso de números aleatórios que emulam uma distribuição de possibilidade na forma triangular, correspondente a definição de parâmetros "fuzzy" em nosso trabalho, ver seção 3.4.3 .

Desta forma, é possível refletir a imprecisão do sistema real, considerando a geração de um valor aleatório que determina o percentual do desvio máximo em relação ao valor central, e a geração de outro valor aleatório, que determina se o desvio será para mais ou para menos.

Em nosso sistema, o desvio máximo é calculado, fornecendo-se um

percentual do valor central do parâmetro. Desta forma, dado um valor central "c" e um percentual de 10%, referente ao desvio máximo de "c", o parâmetro simulado poderá ter valores variando de 0.9.c à 1.1.c .

Também, a simulação de planejamento proposta por Newman e Kempf, a exemplo do modelo de monitoração do SIGT, funciona segundo a operação real. Ou seja, o relógio do sistema é incrementado, progressivamente, segundo os intervalos de varredura, na medida em que as atividades são completadas, sem a realização de retrocessos ou avanços no tempo.

7.6.4 - Planos Universais (M. J. Schoppers [1988])

Concordando com a idéia de conduta competente, Schoppers propõe como técnica de planejamento, a elaboração de planos universais os quais possibilitam ao agente responsável pela execução do planejamento atuar de forma flexível em um domínio com incerteza.

Os planos universais contém, de forma compacta, a representação clássica do universo de planos viáveis, normalmente apresentada sob a forma de um grafo de estados. Entretanto, a parte do plano universal que deve ser executada dependerá, intrísicamente, da situação do domínio em tempo de execução.

- Monitoração x Reação :

Segundo Schoppers, o fundamento básico que suporta esta técnica de planejamento é que :

" Se uma situação S , satisfaz (ou não) uma determinada condição, enquanto voçe tenta alcançar um objetivo G, então a resposta apropriada é a ação A."

Desta forma, podemos observar que no modelo de Schoppers, existe um claro senso de oportunismo condicional, associado ao emprego das ações. Portanto, a conduta do agente associado a esta técnica de planejamento depende, contínuamente, das situações que se sucedem durante o período de execução.

Segundo Schoppers, a conduta deste agente é direcionada para o alcance de objetivos. De fato, a resposta a situações deve seguir uma certa racionalidade . Entretanto, esta racionalidade esta condicionada, geralmente, a consecução de objetivos locais e dependentes da situação em questão.

Para Schoppers, a diferença básica entre a sua técnica de planejamento, e a técnica tradicional de monitoração e replanejamento, utilizada pelo nosso sistema , ver figuras 78 e 79, se deve a inexistência de falhas e replanejamento nos sistemas de conduta reativa.

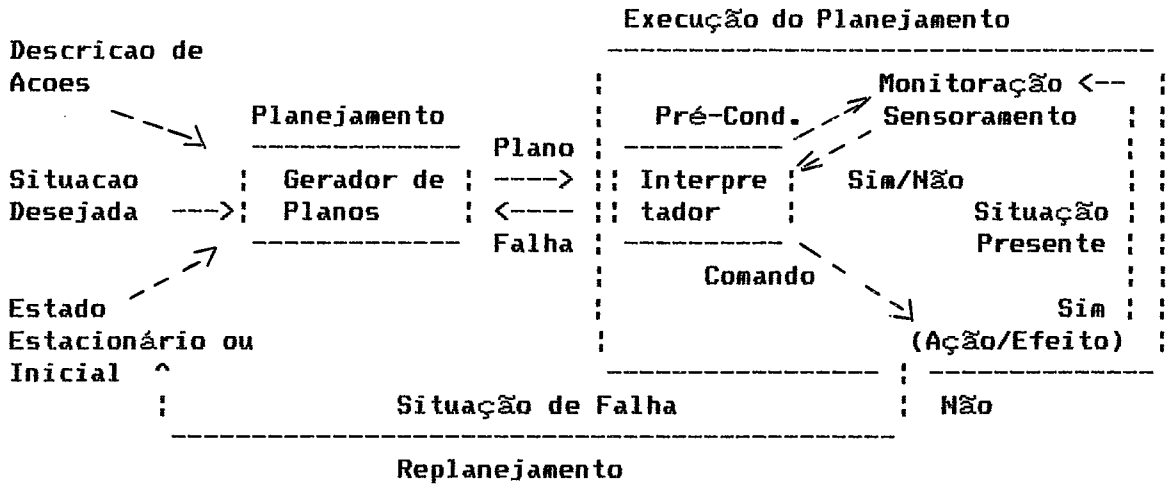


figura 78: monitoração e replanejamento tradicional.

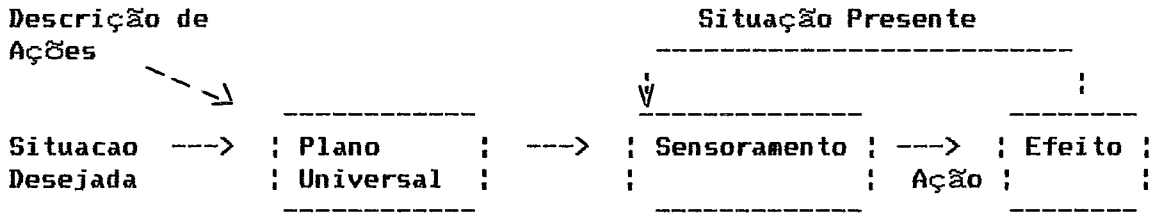


figura 79: plano universal e reação.

Como podemos verificar, no sistema proposto por Schoppers, não existe a descrição de uma situação inicial, e , também, devido a não realização de replanejamento, não existe a possibilidade de falhas. Desta forma, Schoppers propõe uma nova concepção de planejamento,

segundo a seguinte definição:

" Planejamento é a seleção, segundo o alcance dos objetivos, de reações a possíveis situações. "

Relembrando a definição de intuição de R. Dreyfus [1989], podemos considerar planejamento, na visão de Schoppers, como a realização de uma sequência de passos intuitivos envoltos na consecução de objetivos.

Entretanto, sem considerarmos a capacidade deste sistema de planejamento de reconhecer cada possível situação, durante a fase de execução em um domínio complexo em termos combinatório e impregnado de incertezas, queremos lembrar que o processo intuitivo falha. Falha, sobretudo, na consecução de objetivos interativos e conflitantes, que exigem, lógicamente, um pouco de visão a frente , garantida, nos modelos tradicionais, pela realização de uma busca dirigida e racional.

- Interpretação de Planos Universais:

No modelo de Schoppers, ações são consideradas como funções desempenhadas pelo agente (robo) e operadores como os diversos efeitos causados pela execução das ações. Portanto, os operadores dependerão, diretamente, das situações que antecedem o exercício das ações.

Desta forma, o processo de interpretação de planos universais, consiste em determinar uma ação que seja compatível com a situação presente, utilizando, para tanto, uma árvore de decisões associada ao plano universal, que contém um conjunto de condições associadas aos objetivos do problema e as pré-condições existentes. Portanto, a verificação das condições permitirá o desempenho de funções, associadas a ações, que serão avaliadas em tempo de execução.

- Construção de Planos Universais:

O procedimento de construção da árvore de decisões utiliza um método que assume, inicialmente, que as condições do objetivo sejam verdadeiras. De outro modo, o procedimento atua de forma regressiva , assumindo que as condições sejam falsas e verificando as ações

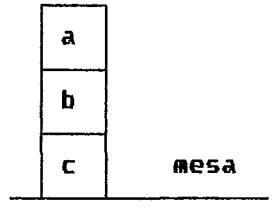
necessárias para o restabelecimento das mesmas.

Em seguida, o procedimento assume que as pré-condições das ações sejam satisfeitas e, do mesmo modo, falsas, atuando de forma recursiva até que haja o restabelecimento das pré-condições. Uma vêz, restabelecida alguma pré-condição, a árvore é percorrida, novamente, até que as condições do objetivo sejam satisfeitas.

Desta forma, é possível gerar descrições parciais do domínio de planejamento, equivalentes a descrição de estados em um universo de planos. Ademais, através do teste de condições , é possível estabelecer um caminho , na árvore decisória, cujo vértice terminal está associado a uma ação desejável , compatível com a situação presente do planejamento.

É possível, também, construir planos universais hierárquicos, através do emprego de ações que constituem outros planos universais. Seja , por exemplo, o problema do mundo dos blocos de se colocar um bloco a sobre um bloco b, e este , por sua vez, sobre um bloco c , utilizando as ações empilha e desempilha, e considerando a situação desejável e as ações descritas na figura 80.

situação desejável:



ações:

empilha (x,y)
desempilha (x,y)
nada

figura 80: exemplo do mundo dos blocos.

Para tanto, tem-se o seguinte plano universal:

```
SOBRE (c,mesa) ?
Sim :  SOBRE (b,c) ?
   Sim:  SOBRE (a,b) ?
      Sim:  nada
      Não: Empilha (a,b)
   Não: Empilha (b,c)
Não: Desempilha (c)
```

Quando se faz a interpretação de planos universais, podemos observar que, quando uma condição se torna um sub-objetivo, na

continuidade do processo, a negação persiste, em todas as situações, até que a mesma seja restabelecida. Ou seja, no sub-plano ou ação empilha(a,b), com o objetivo de restabelecer a condição SOBRE(a,b), a negação do predicado permanecerá sempre.

Portanto, dado que a ação empilha(a,b), requer como pré-condições SOBRE (a,b) e top(a) , significando que o bloco "a" está no topo, o sub-plano universal que restabelecerá a pré-condição SOBRE (a,b) parte do pressuposto que SOBRE (a,b) é falso , reduzindo, desta forma, o trabalho de interpretação.

Neste sentido, o processo de execução de planos que, pelo sensoramento, assume um conjunto de pré-condições é, com certeza, pelo menos tão eficiente quanto o processo de construção de planos universais.

- Considerações:

A crítica que podemos fazer em relação ao trabalho de Schoppers, é que, em domínios complexos, com incerteza, podem aparecer, frequentemente, situações nas quais a árvore de decisões não se aplica, levando, certamente, a uma situação de falha sem solução. Ou seja, não existe, em tempo de planejamento, garantias suficientes de que o programa ou procedimento responsável pela elaboração de planos universais , seja totalmente correto, para todas as possíveis entradas que estarão associadas as ocorrências de situações durante a fase de execução.

Ademais, consideramos o desenvolvimento de planos universais como a realização de uma técnica de planejamento altamente condicional e dependente de suposições realizadas durante a fase de elaboração. Obviamente, dentro deste ponto de vista, não será possível estabelecer, a priori, uma rígida ordenação no estabelecimento de ações, já que, para cada instância ou conjunto de valores assumidos para as condições do plano universal, teremos uma ordenação diferente na execução dos passos do planejamento.

7.6.5 - Tabelas Triangulares

- Introdução:

O uso de tabelas triangulares, segundo N.J. Nilsson [1980], é uma forma eficiente de monitorarmos a execução de planos segundo a filosofia de planejamento do sistema STRIPS, através da observação das pré-condições e efeitos das ações do planejamento, associadas as situações presentes ou reais.

Desta forma, com o uso destas tabelas, é possível observar a ocorrência de algum efeito inesperado, ou verificar a não existência de uma pré-condição , durante a fase de execução das ações.

- Processo de Construção:

Uma tabela triangular consiste de uma matriz triangular inferior construída com as ações, pré-condições e efeitos produzidos no planejamento mais a situação inicial e a situação desejável. Esta matriz possui a coluna inicial, ou coluna 0, associada a situação inicial, e cada coluna restante, associada as ações consecutivas do planejamento. Possui, também, a última linha associada as condições do objetivo do problema e as linhas restantes associadas , da mesma forma, as ações do planejamento.

Desta forma, para cada linha "i" e coluna "j" da matriz , j > 0, a entrada "i,j" corresponde ao literal acrescido pela j-ésima ação, que é pré-condição da i-ésima ação. Neste caso, a última linha se refere aos literais que compreendem o objetivo do problema . Como a coluna 0 se refere ao estado inicial do planejamento, cada entrada "i,0" conterá os literais caracteristicos da situação inicial que sejam pré-condições da ação "i".

Portanto, considerando o planejamento produzido para o problema de sequenciamento de trens, na seção 7.3.2, ou seja : AV(A,2), DESV(B,4), AV(A,3), AV(B,2) e AV(B,1), o objetivo do problema OC(A,3) E OC(B,1) , o efeito das ações, as pré-condições necessárias, e a situação inicial, teremos a seguinte tabela triangular, ver figura 81.

- Monitoração:

O processo de monitoração de planos, a partir de tabelas

triangulares, consiste em determinar de forma iterativa, a partir da última linha , da esquerda para a direita , a ação viável de mais alto grau. Para tanto, para que a i-ésima ação seja viável, verificamos se as pré-condições correspondentes a linha "i" , da coluna 0 até a coluna "j-1" são verdadeiras.

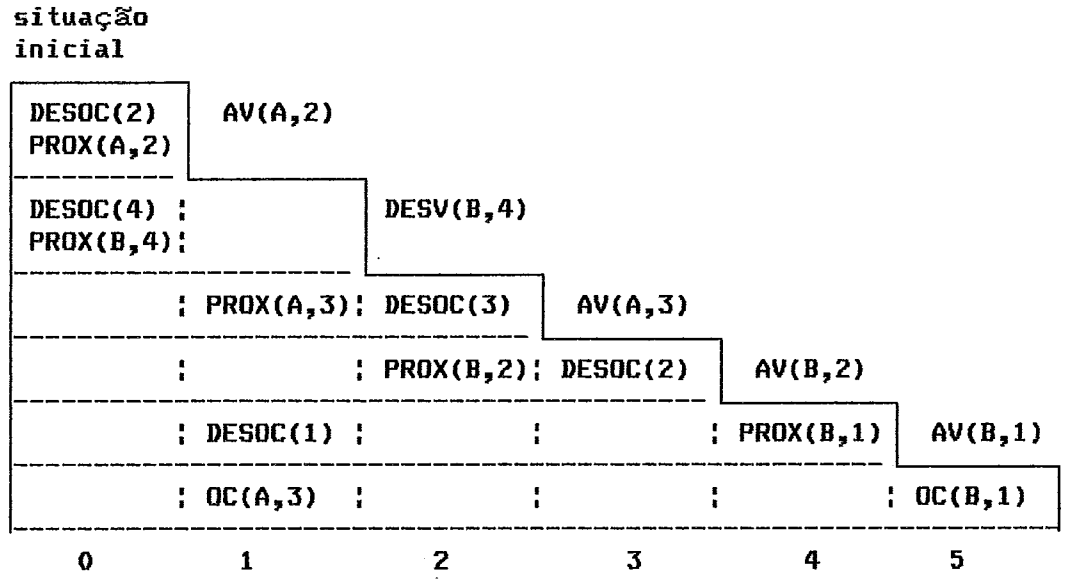


figura 81: representação de planos em tabela triangular.

Neste caso, a realização de replanejamento só será necessária, se não existir nenhuma ação viável na tabela triangular .

Deste modo, é possível, durante o processo de execução, maior flexibilidade no exercício das ações, havendo a possibilidade de se repetir ou desconsiderar a execução de certas ações segundo as pré-condições existentes.

- Considerações:

Segundo Schoppers, a utilização de tabelas triangulares representa uma forma de síntese de planos, que permite uma maior liberdade na escolha de operadores ou ações segundo as situações de ocorrência. Entretanto, dado a rigidez do planejamento tradicional, que especifica não só um ordenamento de ações , como também uma linha de raciocínio , ou uma forma de controle nas ações de modo a alcançar o objetivo do problema, achamos que esta técnica de representação não possui correlação com o desenvolvimento de planos universais.

Portanto, para nós, a exemplo de Nilsson [1980], as tabelas triangulares são formas eficientes de representarmos planos, formados por ações sequencialmente ordenadas, permitindo a monitoração e verificação da viabilidade das ações durante a fase de execução do planejamento.

Neste sentido, o que se determina, realmente, utilizando-se tabelas triangulares, é a ação , a partir da qual, a linha de planejamento deve ser considerada, podendo, naturalmente, ocorrer a supressão ou repetição de passos.

Conforme observações anteriores, podemos observar que, quando não mais se verifica a viabilidade de ações durante a fase de execução , é necessário realizarmos o replanejamento , a partir da situação de falha, contrariando o modelo de Schoppers, que descarta este processo de execução de planos.

A aplicação mais conhecida de tabelas triangulares em sistemas de planejamento se refere ao desenvolvimento do sistema PLANEX (R.E. Fikes, P.E. Hart e N.J. Nilsson [1972]) que monitora a execução de planos STRIPS. Entretanto, este sistema, como já vimos, determina sómente o último ponto do plano no qual a execução pode prosseguir, em função das situações de ocorrência, entretanto, segundo Wilkins [1985], este processo de reinício de planejamento não representa a realização de replanejamento.

- Aplicação no Sistema SIGT:

No sistema SIGT, a utilização de tabelas triangulares , ou uma forma semelhante de representação, que quebre a rigidez do planejamento, pode se mostrar extremamente útil.

Por exemplo, se considerarmos a representação do planejamento proposto na figura 69, na versão simplificada , teremos:

trem1	trem2	seção	ação	instante
A	B	3	2	t

Ou seja, a rotina de execução do sistema interpreta o código da

ação e determina o desvio do trem B , e o avanço do trem A , como o procedimento necessário a solução do conflito de cruzamento que se realizaria no intante "t" na seção$_3$, segundo os seguintes passos:

AV(A,2), DESV(B,4), AV(A,3), AV(B,2) e AV(B,1)

Considerando, agora, a representação de um plano qualquer, composto por um conjunto de ações descritas na forma acima, teríamos uma lista de ações indexadas segundo o tempo de ocorrência dos conflitos , ou seja:

trem1	trem2	seção	ação	instante
A	B	3	2	t
C	A	4	1	t'
...				

Como sabemos, os intervalos de tempo entre conflitos sucessivos é um valor aleatório, que pode assumir valores pequenos, podendo, até mesmo, haver a ocorrência simultânea de conflitos em diferentes pontos da ferrovia.

Portanto, durante o processo de execução, a existência de um ordenamento rígido no estabelecimento de ações , poderia levar a situações de falhas altamente indesejáveis e de fácil solução. Neste caso, se considerarmos o uso de tabelas triangulares, relaxando a restrição de ordenamento imposta pelo planejamento, reduziríamos , totalmente, o número de falhas desta natureza. Conforme observações da seção 7.5.4, sugerimos em nosso sistema um processo de representação semelhante, que examina, sempre, uma lista atualizada de ações mais próximas ao tempo de ocorrência do conflito real ou simulado.

É possível, nas aplicações em tempo real, deixar o procedimento de verificação de viabilidade das ações a cargo do controlador, exigindo, neste caso, uma interação do mesmo com a rotina de execução do sistema de planejamento.

7.6.6 - Recuperação de Erros - SIPE (D.E. Wilkins [1985])

- Introdução:

Wilkins [1985], descreve a capacidade do sistema SIPE (Wilkins [1984]) , quanto ao tratamento de erros durante o processo de execução de planos.

De um modo simplificado, dado uma descrição apropriada, por exemplo, na forma de um predicado, de uma situação inesperada que ocorra durante a fase de execução , e que, possivelmente, afete a viabilidade do planejamento, o sistema SIPE , segundo Wilkins [1985], executa um procedimento de replanejamento, através dos seguintes passos:

1 - Interpreta o significado deste predicado em relação a situação presente;

2 - Determina o problema, ou problemas em questão;

3 - Adota ações de replanejamento de modo a corrigir estes problemas, e

4 - Verifica se as alterações provocadas no plano estão compatíveis com as partes invariantes.

Neste sentido, caso seja necessário a satisfação de novos objetivos, a rotina de planejamento ou de geração de planos é novamente utilizada.

Portanto, podemos considerar o processo de recuperação de erros do sistema SIPE, como um processo de monitoração e replanejamento que, através de ações independentes do domínio, estabelecem no plano original novos objetivos , de modo a corrigir problemas detectados , levando a realização de novo planejamento. Desta forma , o módulo de replanejamento do sistema faz uso de toda potencialidade do sistema de planejamento, como exemplo:

. utilização dos eficientes mecanismos de raciocínio baseados na representação do conhecimento invariante do sistema em "frames", como forma de reconhecer problemas e adotar ações de replanejamento;

. utilização da capacidade dedutiva de operadores, provendo uma razoável solução para o problema de manutenção de verdade.

- Módulo de Replanejamento:

No tratamento de erros de execução, a ocorrência de situações inesperadas, ou situações de falha, são consideradas no sistema SIPE, através da introdução de um predicado que afeta a viabilidade do planejamento em questão. Em termos de representação , esta situação é alcançada através da introdução , na rede procedural, de um novo nó associado ao predicado.

Portanto, os problemas relacionados a translação e fusão de informações, quanto a utilização de sensores (R.P. Bossano [1988]) e, também, quanto a interpretação de cenas como: processamento de imagens, reconhecimento de mensagens, etc. , não são considerados neste sistema.

O módulo geral de replanejamento do sistema SIPE é apresentado, segundo Wilkins, na figura 82.

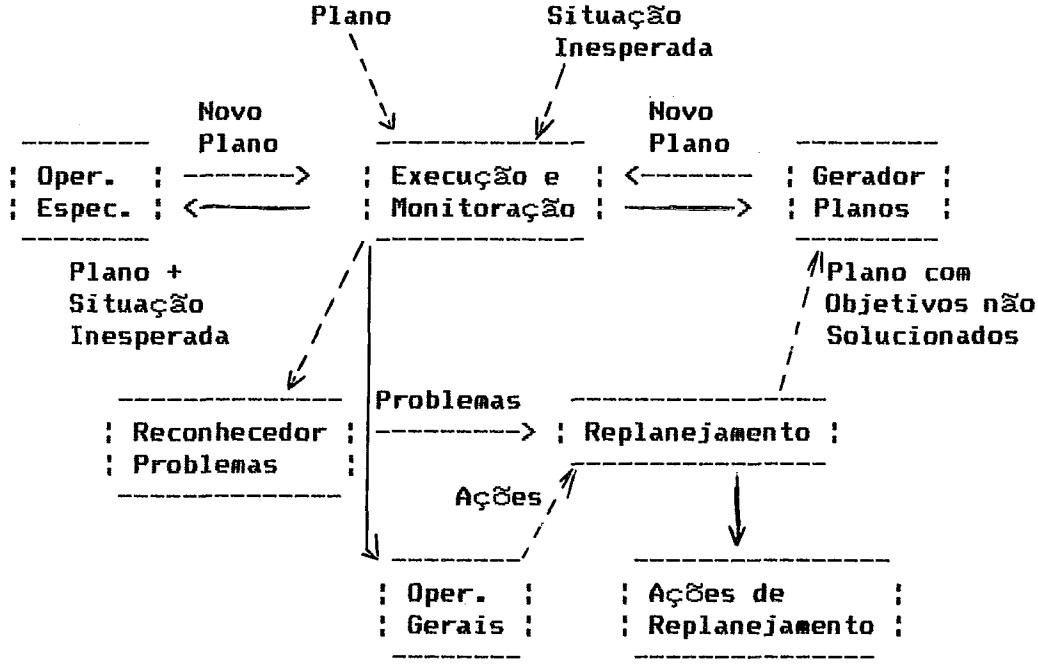


figura 82: módulo geral de replanejamento - SIPE

Como podemos observar, o módulo possui duas linhas básicas de ação. A primeira, já descrita anteriormente, envolve a realização do replanejamento propriamente dito, recorrendo a utilização dos módulos: reconhecedor de problemas, ações de replanejamento e do gerador de planos.

A segunda, consiste na utilização de operadores dependentes do

domínio, que reconhecem situações de falha pela ligação de listas de argumentos e determinam ações apropriadas, cujos efeitos produzem a correção do plano afetado pela situação inesperada.

Existem, neste caso, operadores mais específicos , que são aplicados sem a necessidade de se utilizar a rotina de reconhecimento de problemas, produzindo novos planos. E operadores mais gerais, que são aplicados após a rotina de reconhecimento de problemas, e se apoiam em ações de replanejamento, que são consideradas, posteriormente, pelo módulo de replanejamento.

- Aplicação ao Sistema SIGT:

Como veremos na próxima seção, no nosso sistema empregamos um modelo de replanejamento funcionalmente semelhante ao modelo do sistema SIPE. Neste sentido, ao depararmos com uma situação de falha (situação inesperada) , devemos verificar a possibilidade de efetivarmos medidas de reparo, através da adoção de operadore específicos, que evitem a realização do replanejamento.

Entretanto, os procedimentos que determinam o reconhecimento de problemas e as ações de replanejamento responsáveis pela reavaliação de prioridades, redefinição de objetivos, restabelecimento da situação de falha (estado inicial do replanejamento) , são dependentes do domínio, visando maior eficiência na execução da monitoração e replanejamento em tempo real. Estes procedimentos, também, serão vistos na seção seguinte de nosso trabalho .

- Problemas em um Plano:

Segundo Wilkins, o sistema SIPE considera a existência de seis problemas, após a introdução de um novo nó, associado ao predicado responsável pelo erro, na rede procedural:

1 - Negação pelo predicado, do efeito de uma ação ;

2 - Negação pelo predicado , de uma pre-condição, préviamente estabelecida na rede procedural na forma de um nó, que antecede um operador ainda não executado;

3 - Existência em uma ação de um argumento associado a uma variável não conhecida;

4 - Ocorrência, por parte de uma ação, de um efeito inesperado que produz a negação , para sempre, de uma pré-condição préviamente estabelecida na rede procedural na forma de um nó;

5 - Negação, em tempo de execução, de uma futura pré-condição de um operador, incorrendo na inviabilidade da ação associada, e

6 - Quando todas as pós-condições em paralelo, após um nó junção, não são mais verdadeiras devido ao efeito inesperado de alguma ação.

Para Wilkins, dada a estrutura de representação de planos utilizada pelo sistema SIPE, estes são os únicos problemas que devem ser considerados em tempo de execução, considerando o efeito causado pela ocorrência de situações inesperadas que possam afetar partes presentes e futuras do planejamento.

Finalmente, é abordada a ocorrência do problema de supressão de uma ação, considerando a existência de seus efeitos na situação presente, não se fazendo necessário, portanto, a sua execução.

Para nós, os problemas listados por Wilkins retratam todas as possibilidades de falhas, e são suficientes à execução de planos em domínios favoráveis a realização de planejamento.

Entretanto, em nosso sistema, a deteção de problemas é conduzida de modo mais específico, segundo a probabilidade de ocorrência, e voltada para a utilização, em primeiro lugar , das medidas de reparo. Também, como veremos adiante, não nos preocupamos, diretamente, com o estabelecimento a priori, de pré-condições necessárias a execução futura do planejamento, já que nosso sistema, apresenta, em tempo de execução, uma forma de raciocínio ou inferência do tipo oportunista, voltada para a utilização de informações mais recentes, atualizadas na base de conhecimento.

É evidente, entretanto, que procuramos estabelecer medidas de reparo, que afetem o mínimo possível o planejamento em andamento, postergando , ao máximo, a realização do replanejamento. Utilizamos, até mesmo,medidas corretivas, desde que não afetem os objetivos do problema,

visando a realização de um planejamento forçado.

- Ações de Replanejamento:

Wilkins descreve, também, no mesmo trabalho,um conjunto de ações de replanejamento independentes do domínio, que formam uma base para o módulo de replanejamento do sistema SIPE.

A intenção principal em adotar estas ações de replanejamento, é, em função dos problemas detectados, inserir novos objetivos no planejamento, de modo a corrigir os problemas e preservar o máximo possível do plano original. Neste sentido, o módulo de geração de planos tenta satisfazer os novos objetivos , introduzidos pelas ações de replanejamento, através do processo normal de revisão, incluíndo, possivelmente, a adoção de sub-planos.

As principais ações de replanejamento , apresentadas de forma sucinta, são:

1 - Tentativa de instanciação, se possível, de uma variável diferente, de modo a corrigir a situação de falha, observando se esta ação não provoca problemas futuros;

2 - Inserção de um sub-plano contendo um determinado nó inicial, a partir de outro nó já situado na rede procedural do plano original. Esta ação é tratada, geralmente, como sub-rotina das demais ações;

3 - Inserção de uma condição, em determinado nó, para testar se uma variável, associada a execução da ação, é conhecida ou não, possibilitando a execução de ações condicionais;

4 - Inserção de um nó objetivo no lugar de um nó que determina uma pré-condição que foi negada;

5 - Criação de um novo nó objetivo, associado ao predicado, e, em seguida , inserção na rede procedural segundo o local apropriado (ação 2);

6 - Criação de nós objetivos paralelos para o restabelecimento de pré-condições em paralelo. Em seguida, os nós tambem são inseridos na rede procedural no local apropriado (ação 2), e

7 - Remoção de uma determinada ação, sem inserção de nó, dado a

ocorrência de seus efeitos na situação presente, observando se este procedimento não provoca problemas futuros.

Segundo estas ações de replanejamento, o módulo de replanejamento direciona o processo de criação de objetivos segundo o tipo de problema existente. Por exemplo:

Em primeiro lugar, o sistema procura instanciar uma nova variável, e, em seguida , caso esta ação não seja suficiente, o sistema tenta a quinta ação, inserindo um novo objetivo associado ao predicado.

Por outro lado, se o problema é do segundo tipo, ou seja, existe uma pré-condição préviamente estabelecida que está sendo negada, o sistema tenta, em primeiro lugar , a primeira ação, e, em seguida, tenta, óbviamente, a quarta ação. Se, entretanto, o problema está relacionado ao não conhecimento de uma variável, que seja argumento de uma ação, então, o sistema chama a inserção condicional (terceira ação) , e assim por diante.

- Considerações:

O objetivo principal do módulo de replanejamento do sistema SIPE , é atuar de forma eficiente na execução de planos em domínios como robótica, onde seja possível corrigir, frequentemente, vários erros de execução, conservando grande parte do plano original. Neste sentido, a principal contribuição do sistema se refere a criação de um conjunto de ações de replanejamento associadas a ocorrência de um número limitado de problemas , que formam uma base para um módulo de replanejamento automático.

Entretanto, o principal problema do sistema SIPE, se refere as funções de monitoração e sensoramento, no que tange o acompanhamento de planos e a aquisição de informações provenientes de eventos inesperados. Também, na realização do replanejamento, o sistema SIPE, dada as suas características básicas, não oferece grande capacidade quanto a exploração de um largo espaço de busca, que envolva a realização de várias alternativas de revisão de planos.

7.7 - Execução de Planos no Sistema SIGT

Nesta seção, apresentaremos a descrição do processo de execução de planos no sistema SIGT, considerando a realização da simulação do planejamento. Tratando-se de um sistema de planejamento contínuo e dinâmico, daremos ênfase as questões que envolvem as transformações no sistema ferroviário e a realimentação do planejamento. Também, no que tange ao modelo de monitoração e replanejamento, descreveremos suas rotinas principais, considerando a monitoração de parâmetros, a realização do replanejamento e a utilização das ações de reparo mais comuns.

7.7.1 - Planejamento Dinâmico

- Transformações e Viabilidade do Planejamento:

Em um sistema de planejamento dinâmico, surgem, a cada instante, novas transformações relacionadas, sobretudo, a criação e destruição de novos objetos, bem como, devido a alteração de atributos e valores de objetos existentes.

Na operação de trens, devemos considerar a possibilidade de inserção e deleção de trens ao longo da execução do planejamento, bem como, a mudança de características de classes de trens . Quanto as transformações relacionadas a modificações na malha ferroviária , devemos considerar, apenas, a alteração de características básicas das seções, já que alterações mais profundas do "lay-out" são consideradas, sómente, a nível de decisão estratégica, e não a nível de decisão de controle e planejamento operacional ao qual se adere o nosso sistema.

Sendo, a representação do conhecimento declarativo (constituído de objetos e entidades em nosso sistema) baseado em "frames", foi possível, como já vimos, na implementação dinâmica, manipularmos com eficiência as transformações que se verificaram em tempo de execução (referente ao programa), ao longo do processo de geração de planos.

Entretanto, para a viabilidade do planejamento, é preciso considerar estas transformações, durante a fase de execução dos planos, analizando as consequências causadas, pela introdução de novas

informações, no planejamento original.

- Inserção de Trens:

A inserção de novos trens (não confundir com a realimentação do planejamento) , durante a execução de planos, é provocada, sobretudo, pelo tráfego de trens de serviço não programados e de baixa prioridade. Desta forma, é possível mantermos a viabilidade do planejamento original, considerando a prioridade absoluta dos trens opostos, nos conflitos proporcionados pelos novos trens acrescidos ao tráfego ferroviário.

Esta técnica é, originalmente, considerada na operação real de trens. Contudo, dado a deficiência do processo de comunicação, os trens de baixa prioridade procuram trafegar, sempre, nas janelas, ou espaços vazios da grade de trens, de modo a não interferirem na circulação de trens prioritários. Lógicamente, a adoção de um planejamento prévio, facilitará, bastante, a identificação destes espaços, possibilitando o tráfego de trens de baixa prioridade sem interferência ao movimento.

Quanto a inserção inesperada de trens de alta prioridade, o que ocorre com raríssima frequência, será necessário a realização de novo planejamento. Entretanto, será sempre possível a obtenção desta informação , relativa as características do novo trem, dentro de um período razoável de antecedência, permitindo a realização do planejamento com folga de tempo.

- Deleção de Trens:

A deleção de trens que trafegam pela malha ferroviária é causada, frequentemente, pela ocorrência de acidentes, avarias, ou, também, pelo cancelamento de alguma programação devido a junção (composição conjunta) de dois ou mais trens.

Para efeito da execução do planejamento, poderemos manter a viabilidade original, suprimindo as ações relacionadas aos conflitos que deixarão , óbviamente, de existir. Isto será possível, dado a utilização de tabelas triangulares, ver seção 7.6.5, na representação das ações do planejamento.

É bom ressaltarmos, contudo, que a adoção desta técnica compromete, em certo grau, a otimalidade do planejamento original, dado as mudanças estruturais no problema de sequenciamento de trens. A medida alternativa, consistirá, também, na realização de novo planejamento com a devida antecedência.

- Alteração de Atributos e Valores:

A modificação, durante a execução do planejamento, de valores e atributos de objetos e entidades do sistema ferroviário, através da atualização, em tempo real , da base de conhecimento do sistema SIGT, é tratado como a manutenção de verdade no sistema de planejamento, já tendo sido préviamente discutido na seção 7.4.2 .

Entretanto, não existe uma técnica que permita o tratamento prévio destas transformações, a não ser, a realização de inferência pelo sistema de produção, durante o processo de monitoração do plano, que determinará ou não, em função da viabilidade da ação, a adoção de medidas de reparo ou a realização de replanejamento.

- Realimentação do Planejamento:

Quando consideramos a realimentação do planejamento, estamos referindo , óbviamente, a execução de um planejamento contínuo. Como os planos são elaborados para um horizonte de planejamento limitado, teremos, certamente, que analizar o problema de integração de planos consecutivos e adjacentes.

Neste ponto, queremos esclarecer que nosso sistema é apresentado em nosso trabalho como um sistema único, de abrangência limitada as seções sob controle de um mesmo CTC . Entretanto, conforme a descrição, proposta na seção 2.2.2, o sistema foi concebido de modo a gerenciar o tráfego de trens em diversos sub-sistemas da malha ferroviária, associados aos respectivos centros seletivos e controladores.

Porém, as questões envolvendo a realização de planejamento distribuído, a cooperação de vários agentes e a integração de planos adjacentes não foram considerados neste trabalho, e serão fruto de recomendações para futuros estudos, nas conclusões de nossa dissertação.

7.7.2 - Monitoração e Replanejamento

- Configuração Básica:

Como já verificamos, o processo de execução de planos do sistema SIGT, se caracteriza pela adoção de um modelo de monitoração e replanejamento, a exemplo do sistema SIPE, realizando, entretanto, a reelaboração de planos de forma oportunista, como no modelo de "scheduling" oportunístico de Newman e Kempf, dado a necessidade de resposta ao sistema aplicativo.

Consideramos, também, a existência de um conjunto de operadores , dependentes do domínio, que são responsáveis, na medida do possível, pela realização de ações de reparo que visam preservar o planejamento original. Desta forma, o módulo de monitoração e replanejamento do sistema SIGT, integrado ao módulo responsável pela geração de planos, obedece a seguinte configuração básica, ver figura 83.

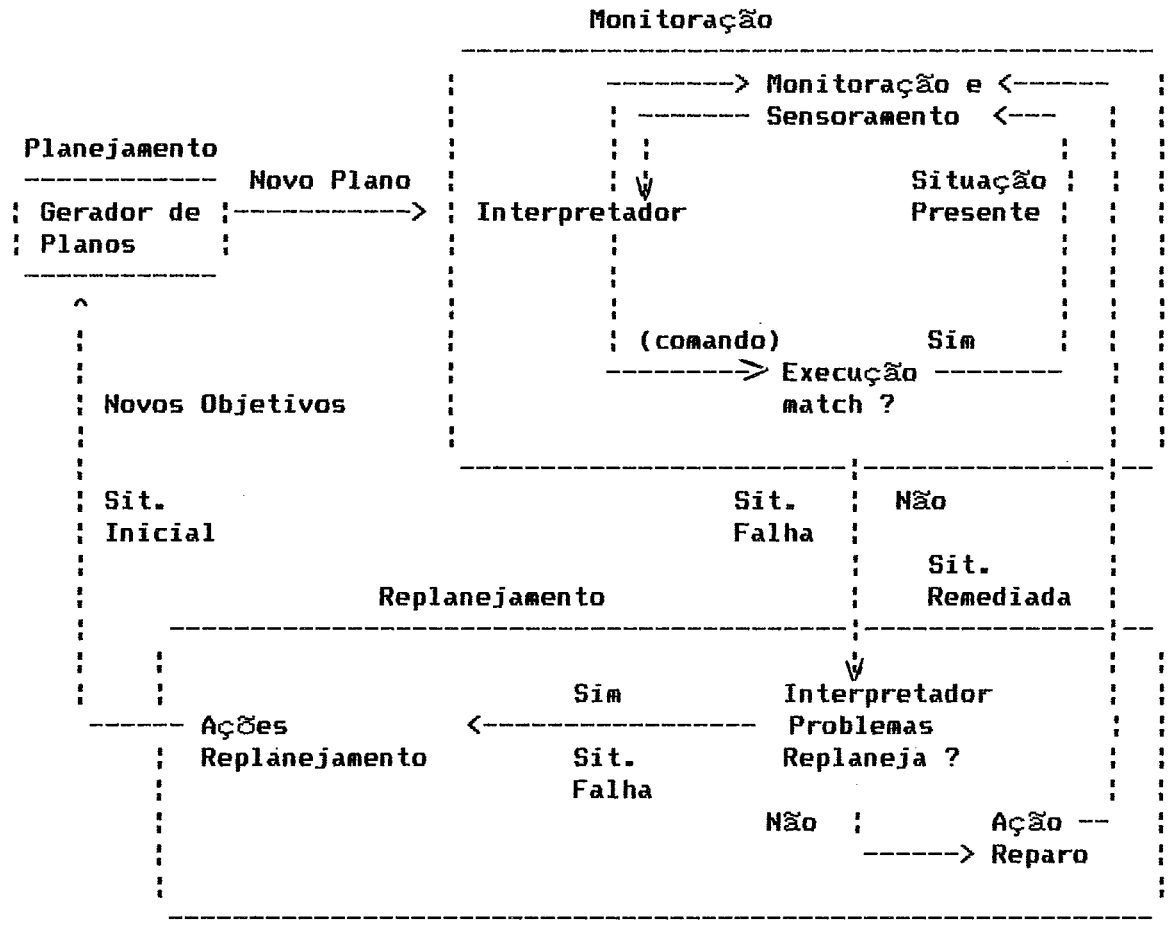


figura 83: módulo de monitoração e replanejamento.

Contudo, o módulo de monitoração e replanejamento, implementado em nosso trabalho, retrata a realização do planejamento de forma simulada, diferindo, portanto, do módulo a ser implementado em tempo real. Entretanto, as diferencas são mínimas e resultantes, sobretudo, da realização de um simulação, tendo como situações presentes, a transição entre estados provocada pela resolução de conflitos projetados.

Neste sentido, as transformações consideradas no sistema de simulação são consequências da imprecisão dos parâmetros relativo ao tempo de percurso dos trens, não sendo observado, portanto, a ocorrência de outros tipos de informações que levem a situações inesperadas.

Entretanto, funcionalmente, os dois módulos são equivalentes, possuindo as mesmas rotinas, garantindo, portanto, a avaliação rigorosa do processo de replanejamento e monitoração do nosso sistema na forma simulada. Em seguida, descrevemos, funcionalmente, cada rotina ou componente do módulo de monitoração e replanejamento implementado para o sistema.

- Interpretador de Conflitos e das Ações do Planejamento:

Como já dissemos na seção 7.5.4, a rotina ou processo de interpretação, dado a existência de um conflito, considera o plano existente e verifica a viabilidade da ação planejada com o conflito em questão, dirigindo, em seguida, um comando a rotina de execução.

- Rotina de Execução:

Caso a ação seja viável, a rotina de execução propõe a execução da ação ao controlador (usuário) e determina uma nova situação em função do efeito desta medida (consideramos, neste caso, a execução obrigatória da ação por parte do controlador). Uma vêz executada a ação, o sistema prossegue a monitoração do plano, observando a aquisição de novas informações (sensoramento) ate deparar-se com nova situação de conflito. Neste ponto, com a devida antecedência, o sistema interpreta, novamente, o plano original e , assim, sucessivamente.

Entretanto, caso não ocorra a viabilidade da ação ou "match", a

rotina de execução chama o módulo de replanejamento que irá interpretar o tipo de problema que provocou a situação de falha.

- Monitoração e Sensoramento:

A rotina de monitoração do SIGT realiza, em intervalos de tempo regulares, conhecido como passo de monitoração, o cálculo dos valores das funções de pertinência de cada trem envolvido no planejamento, bem como, dos atrasos acumulados verificados. Contudo, na implementação em tempo real, devemos calcular, também, a imprecisão verificada nos parâmetros associados aos tempos de percurso direto (sem atrasos) dos trens (na implementação do simulador, não consideramos a monitoração deste último parâmetro, já que o parâmetro de imprecisão é conhecido préviamente, sendo fornecido ao sistema).

O processo de monitoração é contínuo, sendo interrompido, sómente, pela necessidade de resolução do próximo conflito projetado. No sistema de simulação, consideramos a sincronização do processo de monitoração com a rotina de interpretação na forma de co-rotinas, através do estabelecimento de pontos de transferência do controle do programa. Portanto, sendo o relógio do sistema avançado, regularmente, segundo o passo de monitoração, a transferência de volta ao processo de interpretação se dará quando o tempo do relógio estiver imediatamente antes do instante do próximo conflito.

Contudo, na implementação em tempo real, devemos considerar a sincronização de processos verdadeiramente concorrentes, associados ao relógio real. Desta forma, será possível ao interpretador, verificar, permanentemente, a viabilidade das ações, considerando a sucessiva atualização de informações ou sensoramento, segundo as interrupções existentes, e, ao mesmo tempo, executar o processo de monitoração de parâmetros segundo intervalos regulares.

Neste sentido, queremos ressaltar que a função de sensoramento não foi considerada na implementação simulada, pois as situações presentes, como já dissemos, são consideradas, sómente, sob o efeito das ações de planejamento que provocam as transições entre estados. Desta forma , a simulação do movimento é conseguida através da utilização de uma tabela de tempos com parâmetros "fuzzy" ou nebulosos, que

representam a imprecisão verificada no sistema. Ou seja, podemos dizer que a tabela de tempos, sob o efeito das ações de planejamento, simulam a aquisição de informações (posição dos trens) conflito a conflito.

- Interpretador de Problemas:

Esta rotina foi pouco desenvolvida em nosso sistema, sendo, essencialmente, voltado para a interpretação de situações de falha no sentido de permitir a adoção de ações de reparo.

Como exemplo, vamos analisar uma situação típica, responsável por grande parte das falhas ocorridas durante a execução do planejamento. Dada a imprecisão dos parâmetros associados aos tempos de percurso dos trens, é muito comum a posição do conflito original, ver figura 84, não corresponder a posição do conflito real ou simulado. Neste sentido, podem ocorrer dois tipos de variações:

1 - A posição do conflito permanece dentro da mesma seção. Neste caso não haverá problemas quanto a viabilidade da ação do plano;

2 - A posição do conflito não permanece dentro da mesma seção. Neste caso, teremos duas situações, exemplificadas na figura 85 , que causam a inviabilidade da ação do plano original.

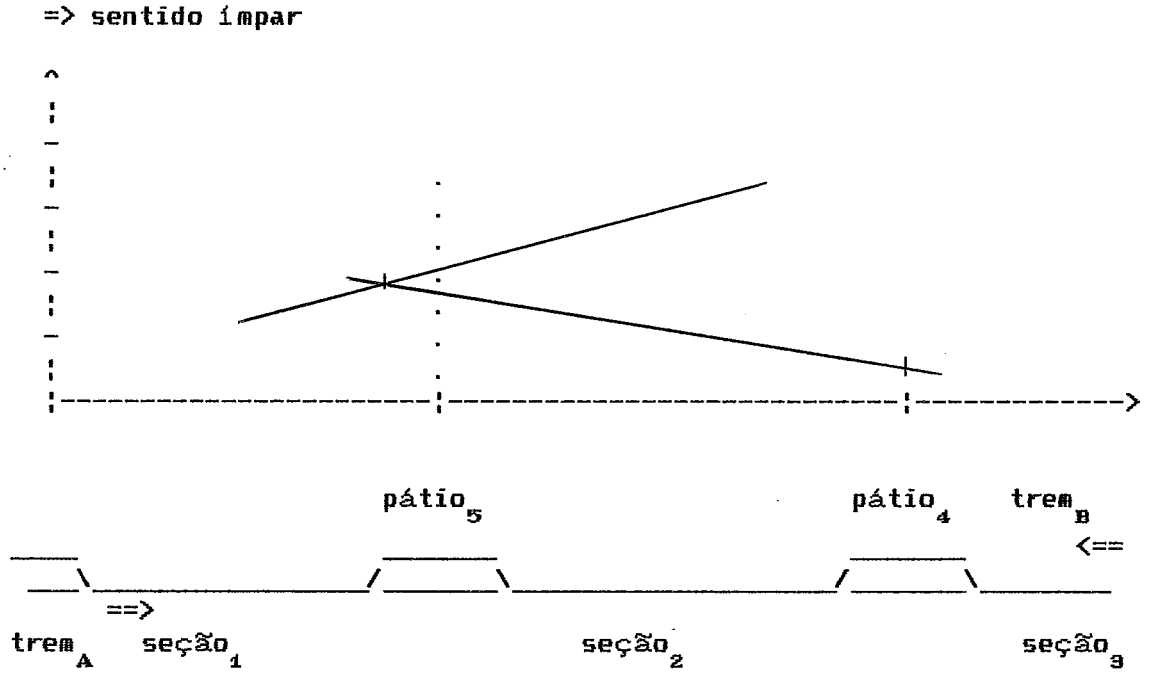


figura 84: conflito original.

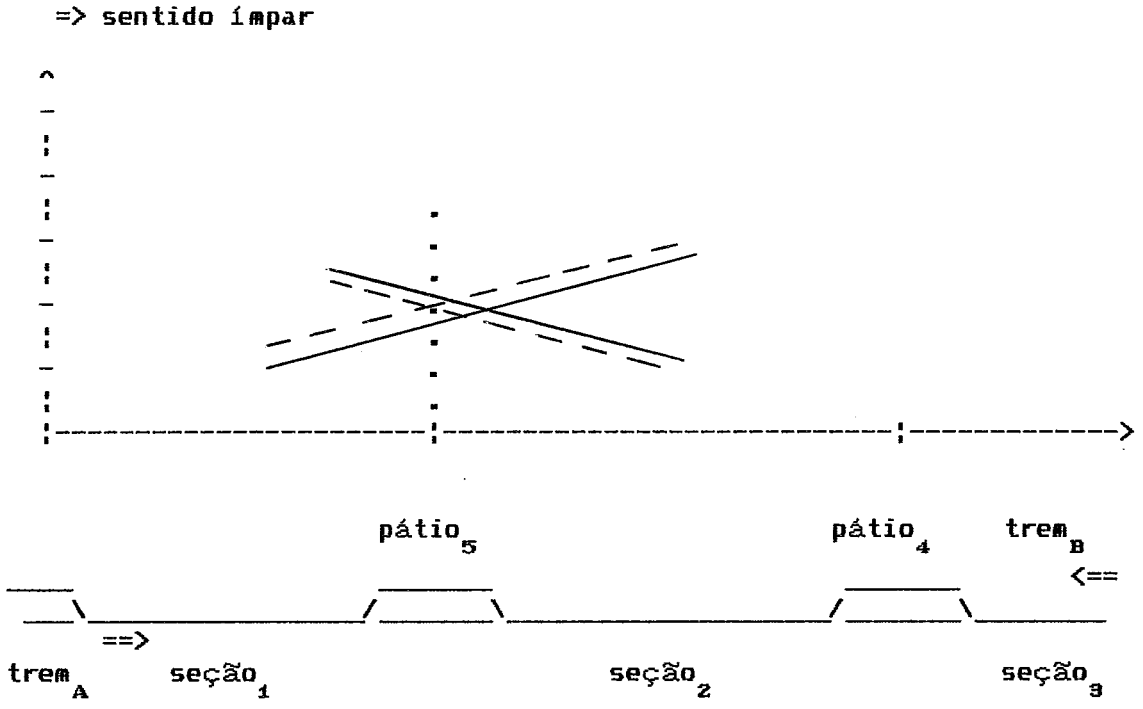


figura 85: novo conflito.

Como podemos observar, no segundo caso, durante a execução do planejamento poderá ocorrer (linha tracejada), a ausência de conflito provocada pela realização do cruzamento de trens em movimento. De outra forma (linha inteira) teremos a ocorrência do conflito na seção adjacente, implicando na inviabilidade da ação do plano original. Observe que, quanto mais próximo a posição do conflito original estiver do pátio adjacente, mais provável será a ocorrência da variação de seção.

Na primeira situação, com a ocorrência do cruzamento com trens em movimento, teremos a supressão da ação de planejamento associado ao conflito original. Este procedimento, como já vimos, será devidamente considerado pela rotina de execução, pelo fato de utilizarmos tabelas triangulares ou uma estrutura semelhante para a representação das ações de planejamento.

Entretanto, caso o interpretador de problemas detecte a ocorrência da segunda situação, será necessário a adoção de uma ação de reparo para recuperar o planejamento original.

- Ações de Reparo:

Para repararmos a situação de falha, teremos que considerar a ação do plano original. Neste caso, se a ação determinava a parada do $trem_B$ no $pátio_5$, teremos, na situação real, a possibilidade de desviarmos o $trem_A$ para o $pátio_5$ ou o $trem_B$ para o $pátio_4$.

Entretanto, se os trens A e B forem de mesma classe, ou se o $trem_A$ for um trem de menor comprimento que o $trem_B$, podemos, a principio, trocar os atrasos dos trens referentes a decisão do conflito de cruzamento, determinando a parada do $trem_A$ no $pátio_5$.

De outra forma, a parada do $trem_B$ no $pátio_4$, apresentaria as seguintes incoveniências:

1 - Teríamos que verificar, em tempo de execução, a viabilidade da parada do $trem_B$ no pátio. Para o $trem_A$, seria necessário verificar se o pátio de cruzamento comporta o mesmo, sómente se o seu comprimento fosse maior que o comprimento do $trem_B$;

2 - O deslocamento do conflito para a seção adjacente, a direita, no sentido do $trem_A$, pode ser provocado por dois motivos: adiantamento do $trem_A$ ou atraso do $trem_B$. Portanto, será preferível, como medida corretiva, parar o $trem_A$ ao invés de pararmos o $trem_B$, já que a medida contrária provocaria mais atraso para o $trem_B$, e a medida proposta proporciona a diminuição do adiantamento do $trem_A$;

3 - Nas condições em que se apresenta a situação de falha, o atraso do $trem_A$, ocasionado pela sua parada, será muito inferior ao atraso do $trem_B$, caso optassemos pela medida contrária. Neste sentido, podemos verificar que esta ação de reparo atende a uma heurística básica do sistema decisório.

Desta forma, a parada alternativa do $trem_B$, no $pátio_4$, só será considerada, se não houver a possibilidade física ou operacional, de pararmos o $trem_A$.

É interessante observarmos, que esta ação de reparo, que determina a parada do $trem_A$, será tanto mais eficiente quanto mais

próximo o conflito estiver do pátio de cruzamento . Também, como medida corretiva, o restabelecimento do plano original será tanto maior quanto mais próximos forem o tempo que separa os dois conflitos (adiantamento do $trem_A$) e o tempo resultante da espera no pátio de cruzamento, ver figura 86.

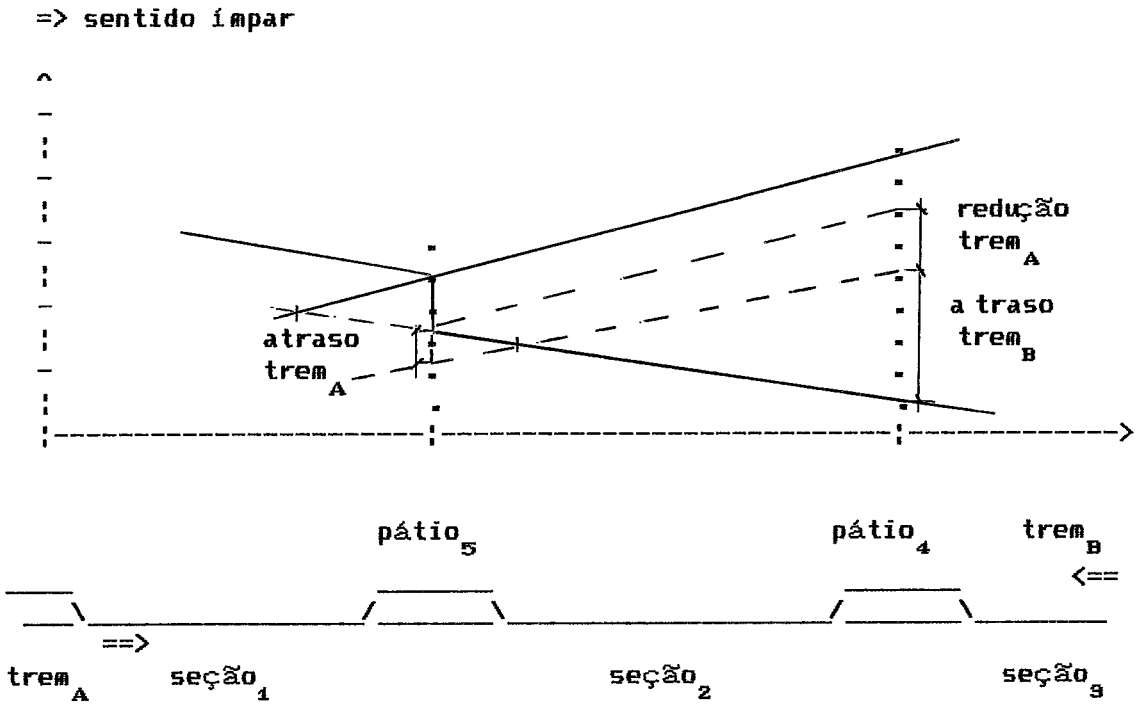


figura 86: relação entre tempos na parada do trem A.

Outro problema, facilmente detectado, ocorrerá quando não houver, no plano original (linha tracejada), uma situação de conflito dado a existência de cruzamento com trens em movimento. Neste caso, na operação real, teremos o surgimento de um novo conflito (linha inteira) , conforme a situação da figura 87.

Neste caso, a ação de reparo será, obviamente, a que determina a parada do trem que estiver mais próximo ao pátio de cruzamento (no exemplo, o $trem_A$), desde que seja verificada a viabilidade desta medida, seguindo, assim, a mesma heurística do procedimento anterior.

Contudo, se a projeção do novo conflito estiver associada a um ponto próximo ao meio da seção, a ação de reparo deve determinar a parada do trem que estiver com menor atraso acumulado (outra heurística do sistema decisório) ou determinar a realização do replanejamento.

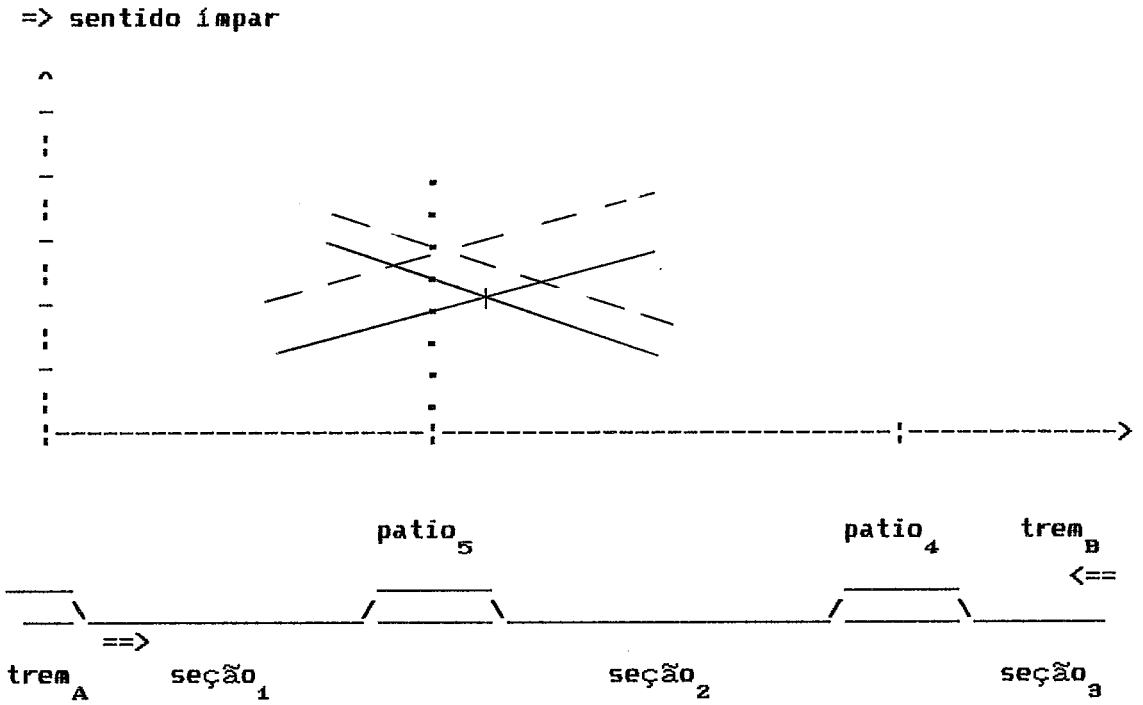


figura 87: ocorrência de novo conflito.

Na figura 88, podemos observar como o sistema SIGT considera o funcionamento do módulo de replanejamento, segundo a situação exemplificada na figura 87 . Inicalmente, o problema detectado é mostrado textualmente. A seguir, o sistema oferece as opções de operadores (voluntárias) ou a possibilidade de ação de reparo (involuntária) em função do problema detectado. Caso não seja possível a adoção de uma medida de reparo, o sistema recomenda a realização do próprio replanejamento.

- Ações de Replanejamento:

No sistema SIGT, a exemlo do sistema SIPE (D.E. Wilkins [1985]) , as ações de replanejamento, visam, sobretudo, o redirecionamento do planejamento a partir das situações de falha .

Para tanto, o sistema adota um conjunto de procedimentos, dependentes do domínio, que envolvem:

. Redefinição de objetivos e

. Reconstrução das tabelas de tempo ou do estado incial do planejamento.

Uma vêz realizado estes procedimemtos, o sistema invoca o módulo gerador de planos que determinará a criação de um novo plano de ações.

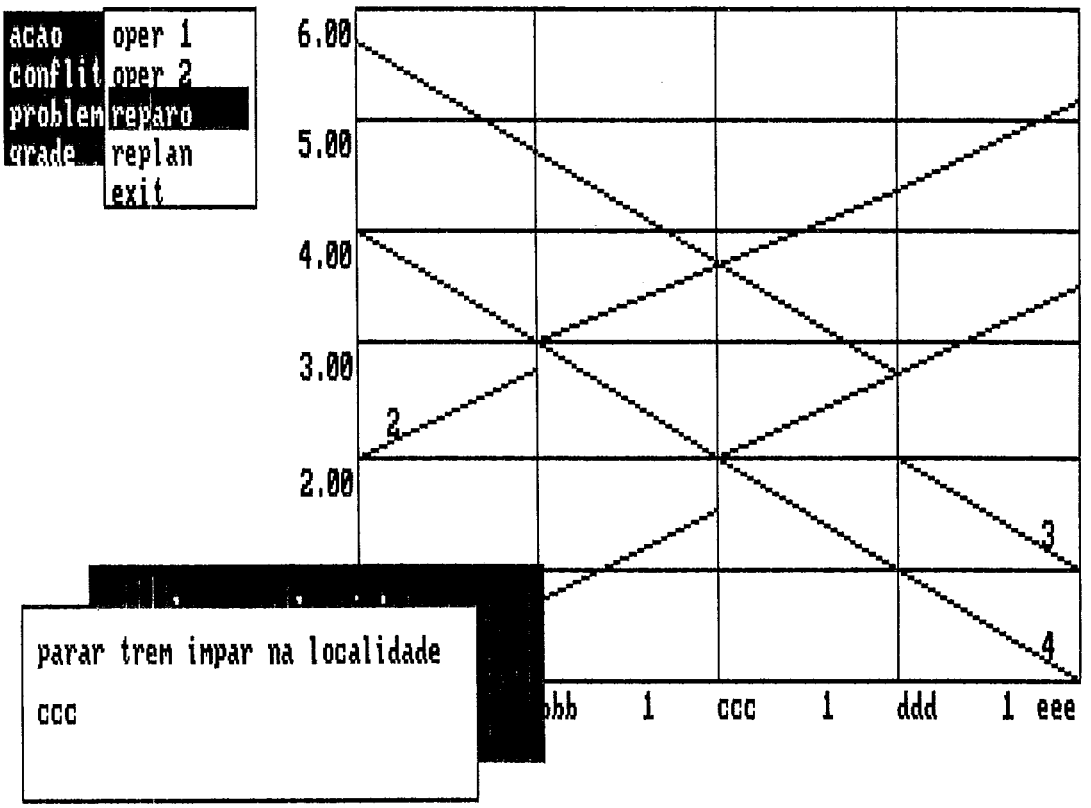


figura 88: tela do módulo de replanejamento.

A redefinição de objetivos é realizada de forma compensatória e interativa, alterando-se os suportes ou intervalos das funções de pertinência , ver seções 3.5.1, 3.5.2 e 3.5.3, através de ceeficientes multiplicativos.

Estes coeficientes são calculados definindo-se um vetor de fatores de reajustamento na forma de percentuais, ver seção 3.5.5, associados aos níveis de satisfação. Desta forma, para cada objetivo, determina-se um fator em função dos tempos acumulados de cada trem ou grupo de trens, associados aos respectivos valores das funções de pertinência, ou seja :

Para $k = 1$, fator $\leftarrow fat_1$.

Para $k = 2,...,5$, Se $B_{k+1} < t \leq B_k$ Então

$$fator \leftarrow fat_k + ((fat_k - fat_{k-1}).(t - B_{k+1}))/(B_k - B_{k+1}).$$

E, finalmente:

Se $t > B_5$ Então , fator <- fat_5 ,

onde :

"t" se refere ao valor da função de pertinência associado a cada objetivo;

B é o vetor de níveis de satisfação, e

fat é o vetor de percentuais de reajuste.

Portanto, para cada fator , temos calculado um índice percentual que, multiplicado ao atraso original, definirão novos intervalos ou suportes para as funções de pertinênncia. Estes novos suportes serão maiores, menores ou iguais aos intervalos anteriores, dependendo, lógicamente, dos valores das funções de pertinência que darão uma idéia do grau de satisfação dos objetivos.

Convencionalmente, para valores de "t" acima do nível B_3 , consideramos um aumento no novo atraso, e para valores inferiores, consideramos um decréscimo no novo atraso.

Uma vez determinado os novos atrasos, modificamos, da mesma forma, as prioridades dinâmicas dos trens, e recalculamos as funções de pertinência , redefinindo os objetivos do problema.

A redefinição dos critérios que determinam o cálculo dos atrasos originais, bem como a modificação das prioridades estáticas dos trens , serão possíveis, sómente, na realização de novo planejamento no processo normal de realimentação que determina a integração de planos consecutivos.

Uma vêz redefinidos os objetivos do problema, o sistema determina a reconstrução da tabela de tempos e a construção de um estado inicial para a realização do replanejamento.

A nova tabela de tempos considera a tabela de tempos real ou simulada até o instante de ocorrência da situação de falha, condicionada, provavelmente, a existência de um conflito sem solução. Desta forma, recalculamos, sómente, os elementos da tabela a partir deste instante, levando-se em conta os tempos de percurso originais.

Neste sentido, o cálculo das funções ou a avaliação dos objetivos continua sendo considerado, a partir do instante inicial do planejamento, possibilitando a perseguição de metas e atendimento aos critérios estabelecidos inicialmente, devido ao efeito acumulado das ações do planejamento e da movimentação dos trens .

Dentro deste ponto de vista, consideramos a redefinição dos objetivos como um processo de redirecionamento do planejamento de modo que o efeito das ações planejadas expressem com maior eficiência o alcance dos objetivos segundo as metas originais. Desta forma, os atrasos em excesso verificados em alguns trens, são recuperados com maior rapidez, sendo, entretanto, contrabalançados pelo maior atraso de trens que possuíam uma performance superior.

Uma vêz reconstruída a tabela de tempos, é gerado o estado inicial do novo planejamento, com base nas informações do estado real ou simulado, associado a situação de falha.

7.7.3 - Simulação do Planejamento

- Objetivos:

A exemplo do sistema de Newman e Kempf, desenvolvemos para o SIGT um modelo de simulação do planejamento que permite a execução de planos de forma simulada , em um computador ,considerando a realização da monitoração e replanejamento, na forma descrita anteriormente.

É indispensável, ressaltarmos as vantagens proporcionadas pela possibilidade de simularmos em um computador a execução do planejamento. Entretanto, queremos observar que esta proposta de implementação, visa, em nosso trabalho, avaliar, principalmente, a viabilidade do processo de planejamento e replanejamento, que se verifica durante a execução de planos, procurando refletir, ao máximo, a operação real e o gerenciamento do tráfego de trens.

Desta forma, queremos esclarecer que algumas funções relacionadas, principalmente, ao sensoramento ou aquisição de dados, projeto de interface e transferência de dados em tempo real, não são

considerados na simulação de nosso modelo de monitoração e replanejamento.

- Ambiente de Simulação:

Para que a simulação do planejamento fosse possível, determinamos a criação de um ambiente de concorrência, onde se destacam três processo principais, que compreendem o modelo proposto na figura 83.

1 - Um processo responsável pela simulação de conflitos, que abrange, também, as funções de interpretação e execução de ações;

2 - Um processo responsável pelo replanejamento, abrangendo as rotinas responsáveis pela interpretação de problemas, ações de reparo e ações de replanejamento. Este processo está associado, óbviamente, ao sistema gerador de planos ou processo principal, e, finalmente,

3 - Um processo responsável pela monitoração, que efetua o controle dos parâmetros associados ao movimento de trens. Estes parâmetros, que na implementação em tempo real seriam obtidos por sensoramento, são inferidos, a cada intervalo, de uma tabela de tempos simulada.

A sincronização destes processos é baseada na transferência de controle entre co-rotinas, utilizando-se uma implementação de semáforos com contadores (G.Ford e R.S. Wiener [1985]). Neste sentido, a implementação real de processos concorrentes baseada na reprogramação de processos por interrupções de um relógio interno (D.A. Sewry [1984a,b]) não se faz necessária, devido as considerações já feitas anteriormente.

- Simulação de Conflitos, Interpretação e Execução de Ações:

Este processo, denominado SimuPlan (simulador de planos) , compreende, basicamente, um "loop" que, a cada iteração, projeta a ocorrência de um conflito, com base na tabela de tempos construída segundo a imprecisão do sistema. Em seguida, o processo interpreta o conflito presente e a ação planejada, verificando a sua viabilidade, e enviando um comando a rotina de execução.

Na fase de interpretação, como sabemos, é verificada a viabilidade da ação de planejamento proposta em relação ao conflito simulado, sendo possível, ver seções 2.2.4 e 7.5.4 , a visualização gráfica da grade parcial de trens, bem como da forma textual do tipo de conflito e da ação de planejamento proposta.

Caso não haja a ocorrência de ligação ou "match" (ação não é viável) o processo transfere o controle do programa para o processo de replanejamento, o qual , irá propor uma ação de reparo , ou um novo planejamento.

Uma vez determinada uma ação viável, a rotina de execução determina a sua utilização. Esta execução, provocará a transição de estados através da aplicação de um determinado operador no estado simulado, gerando uma nova situação presente e um novo conflito projetado.

- Processo de Monitoração :

Na simulação do planejamento, a transição entre estados é feita de forma gradativa , através da transferência de controle, após a execução da ação , do processo anterior para o processo de monitoração, denominado MoniPlan (Monitoração do Planejamento).

Este processo, responsável pelo avanço do tempo, efetuará, como já vimos, o controle de parâmetros segundo um passo estipulado, até que o tempo de simulação anteceda o instante de ocorrência do último conflito projetado. Neste instante, o controle do programa volta ao processo anterior, que reiniciará sua execução, tendo em mão um novo conflito e uma nova ação de planejamento. Obviamente, o processo de simulação avança até que o tempo de planejamento tenha esgotado, ou não se verifique mais a ocorrência de conflitos.

- Sincronização dos Processos:

Em seguida, apresentamos uma descrição básica dos três processos responsáveis pela simulação do planejamento, destacando os pontos de trensferência de controle e as rotinas principais. Para maior esclarecimento, ver também o apêndice II.

```
Processo SimuPlan;

  RotinaExecução ( comando );
  Início
    Se  (comando = ação_não_viável) Então
        transfere ( Replanejamento );
        aguarda ( Replanejamento );
        Viabilidade_da_Ação;
        RotinaExecução (comando)
    Senão
        Executa_Ação
    Fim
  Fim RotinaExecução;

Início
  acabou <- FALSE;
  Loop
    aguarda ( MoniPlan );
    Retira_Conflito;
    Interpretação_do_Plano;
    Viabilidade_da_Ação;
    RotinaExecução (comando);
    Transição entre Estados;
    Projeta_Conflito;
    Se (não há mais conflito) Então
      acabou <- TRUE
    Fim;
    Se não(acabou) Então
      transfere ( MoniPlan )
    Senão
      termina processos
    Fim
  Fim
Fim SimuPlan;


Processo Replanejamento;
Início
  Loop
    aguarda(SimuPlan);
    Interpretação_Problema;
    Se (há ação de reparo) Então
      transfere(SimuPlan)
    Senão
      Redefine_Objetivos;
      Reconstroi_Estado_Inicial;
      GeradorPlanos ( Objetivos, Estado Inicial);
      transfere(SimuPlan)
    Fim;
  Fim
Fim Replanejamento;


Processo MoniPlan;
Início
  Loop
    aguarda (SimuPlan);
    Avalia_Parâmetros;
```

```
      Imprime_Tela;
      Se (tempo_simulação > instante_PróxConflito)  Então
        transfere(SimuPlan)
      Fim;
      Avança tempo_simulação;
      Se tempo_simulação > tempo_planejamento Então
        acabou <- TRUE;
        transfere(SimuPlan)
      Fim
    Fim
Fim MoniPlan;
```

Capítulo XIII
Conclusões e Recomendações

8.1 - Considerações Finais:

É notória a necessidade de otimização e automação das atividades e servicos relacionados ao sistema ferroviário, principalmente, no que se refere ao gerenciamento do tráfego ferroviário ou tráfego de trens. Com a melhoria deste gerenciamento, e possível aumentarmos a capacidade de transporte nas ferrovias, sem que, para tanto, sejam feitas investimentos vultosos em alterações de sua planta física ou na aquisição de material rodante e de tração.

Neste sentido, combinando técnicas de Inteligência Artificial, Teoria de Sistemas, Micro-Computação e Pesquisa Operacional, desenvolvemos um sistema inteligente que permite, sob a ótica do planejamento, a realização do gerenciamento do tráfego ferroviário de forma eficiente, visando, sobretudo, a automação parcial deste serviço na ferrovia.

Seguindo o raciocínio exposto em nosso trabalho, achamos que , primeiramente, nosso sistema foi bem sucedido no aspecto funcional e organizacional, atendendo, plenamente, as solicitações e aos requisitos básicos de capacidade relatados ao gerenciamento do tráfego ferroviário. Ademais, tendo sido desenvolvido com base em justificativas e hipóteses bem fundamentadas, a implementação do sistema SIGT constitui um excelente programa de computador, capacitado a realização do planejamento operacional, tático e estratégico na ferrovia.

Quanto ao desenvolvimento teórico do sistema de planejamento, queremos ressaltar os aspectos referentes a elaboração de uma arquitetura inteligente, considerando os processos universais de busca, raciocínio, satisfação de restrições, abstração e decomposição , empregados na solução de problemas e no desenvolvimento de sistemas de programação sob o enfoque de múltiplos paradígmas. Neste sentido, queremos destacar a visão multi-disciplinar do nosso processo construtivo, que dirigiu o desenvolvimento de nosso sistema e possibilitou a solução do problema de sequenciamento de trens associado a geração de planos de horários na forma mais realista.

Tal visão e esforço de trabalho, permitiu que relacionassemos várias áreas de conhecimento em Engenharia de Sistemas, como Teoria de "Scheduling", Programação Multi-Objetiva, Otimização "Fuzzy", Solução de Problemas, Buscas Heurísticas, Sistemas Simbólicos, Bases de Conhecimento, , Sistemas de Produção , Arquiteturas Inteligentes, Simulação, e, finalmente, Planejamento e Execução de Planos em Inteligência Artificial, em um esforco conjunto no sentido de formar uma base sólida de representação e utilização do conhecimento associado ao domínio de nossa tarefa, com o claro objetivo de prover capacidade ao nosso sistema.

Este enfoque, justifica, a princípio, o fato de implementarmos uma arquitetura, em grande parte, dependente do domínio, e dirigida pelo uso do conhecimento intensivo, sacrificando a generalidade em troca da eficiência necessária. Entretanto, consideramos nossa proposta de implementação, como uma metodologia a ser utilizada no desenvolvimento de sistemas inteligentes, com aplicação em diversos problemas de planejamento e "scheduling".

8.2 - Contribuições :

Quanto aos resultados teoricos obtidos, queremos ressaltar as principais contribuições, baseadas e fundamentadas no texto, que se apresentaram no decorrer de nossa pesquisa:

1 - A extensão de um modelo matemático com a aplicação de técnicas de Inteligência Artificial e consequente desenvolvimento de um sistema simbólico . A partir da utilização das teorias de "scheduling", programação multi-objetiva e programação "fuzzy" , foi possível desenvolver uma representação espaço-estado para o problema, operadores, heurísticas e estratégias de controle, que conduziram a soluções não-dominadas e plenamente satisfatórias;

2 - A extensão do sistema simbólico a partir da utilização do conhecimento intensivo e consequente desenvolvimento de uma arquitetura inteligente. Esta arquitetura proporcionou a obtenção de resultados ainda mais expressivos, quanto ao cumprimento de um conjunto de novas restrições, e , tambem, algoritmos mais eficientes, que implicaram na redução do esforço computacional necessário a solução do problema;

3 - O desenvolvimento de um sistema de programação em múltiplos paradigmas, permitindo uma forma de organização e projeto que integrou técnicas de programação procedural, programação baseada em regras e programação orientada para objetos, possibilitando a consolidação de um ambiente de programação altamente flexível, modular e amigável;

4 - O desenvolvimento de um sistema baseado em conhecimento que combinou a utilização de "frames" com a utilização de um sistema de produção . A implementação deste sistema se fundamentou na filosofia de tipos abstratos, e abstrações funcional, tendo como finalidade, suportar a realização de inferências de forma eficiente e permitir o acesso e a manipulação do conhecimento declarativo por parte de outras rotinas do sistema SIGT;

5 - O desenvolvimento de um modelo de simulação, com base na arquitetura inteligente , que permitiu alto grau de paralelismo na realização de eventos, e maior flexibilidade nos mecanismos de controle e programação, sendo de fundamental importância para o aprendizado do sistema segundo o estudo e a análise de casos;

6 - A associação entre o processo construtivo do sistema gerador de planos, e a teoria de planejamento em Inteligência Artificial. Particularmente, queremos destacar a forte relação com as técnicas mais eficientes de construção de planos, como a estratégia "least-commitment" do sistema NOAH, , a estratégia "constraint-posting" do sistema MOLGEN, a interpretação do criterio da verdade no sistema TWEAK, e o raciocínio sobre recursos no sistema SIPE.

Consideramos, tambem, de grande contribuição, os estudos sobre dominância entre planos no sistema SIGT, relacionado planejamento e otimização, a execução de ações paralelas na presenca de um unico agente e, por fim, a proposta de organização.

No que se refere a realização do planejamento sob incerteza e planejamento temporal, destacamos :

1 - O enquadramento do sistema segundo a realização de planejamento, monitoração e replanejamento, face ao grau de previsibilidade e reatividade do sistema ferroviário, e as considerações

sob as fontes de incerteza existentes;

2 - A manutenção de verdade no sistema, apoiada na utilização de informações dependentes e na realização de inferências oportunistas;

3 - A estrutura de representação temporal com base na tabela de tempos e o gerenciamento de relações temporais com base nas restrições disjuntivas e de sequenciamento do problema de "scheduling";

4 - A capacidade de raciocínio temporal sob incerteza baseada na recuperação de informações e na realização de calculo de intervalos com base na teoria pontual, e, finalmente,

5 - O atendimento as pre-condições necessárias ao estabelecimento e a representação de ações e, consequente, exercício do planejamento em tempo real.

Finalmente, no que tange a execução de planos, como não foi possivel a obtenção de resultados práticos a partir de implementações em tempo real, queriamos considerar o desenvolvimento do módulo responsável pela simulação do planejamento. O objetivo básico deste módulo, foi permitir a execução do planejamento segundo emulações ou operações simuladas.

A simulação do planejamento, possibilitou a monitoração, e consequente avaliação de planos, considerando a fonte principal de incerteza do sistema ferroviario, relativo a imprecisão das entradas da tabela de tempos. Permitiu, tambem, a realização do replanejamento incluindo a adoção de algumas medidas de reparo, tendo contribuido de maneira decisiva, para uma pre-viabilização do sistema SIGT em tempo real, segundo um modelo de monitoração e replanejamento.

8.3 - Recomendações

É evidente, que o funcionamento pleno do sistema, em condições reais de operação, envolveria, certamente, muitos anos de esfôrço e pesquisa. Neste sentido, consideramos, em nossas recomendações, algumas

sugestões, as quais poderao servir, futuramente, como temas de estudo, no sentido de dar continuidade ao desenvolvimento teórico e a implementação final do sistema SIGT.

Portanto, faremos, a seguir, um relato das deficiências ou restrições do sistema, origindo, por consequência, indicativos valiosos para a continuidade do trabalho.

Em primeiro lugar, vamos considerar as principais restrições do sistema de planejamento, na forma da implementação atual :

1 - Restrição ao modelo de implementação em relação aos recursos de tempo e memória. Esta restrição se deve, sobretudo, a utilização do ambiente de programação Modula-2, Logitech, versão 2.0, baseado no sistema operacional DOS 3.3 e processador Intel 80286, e, ultimamente, na versão 3.0 e processador Intel 80386SX, ambas, sem o coprocessador numérico.

Apesar de considerarmos, que o uso deste ambiente, com o auxílio do coprocessador 80387SX, e com a representação de dados em memória virtual (expansão de 1.4 MB), satisfaz, plenamente, a implementação atual do sistema, sugerimos que sejam desenvolvidos, novos modelos de implementação , na linguagem Modula-2, ou em novas extensões como Modula-2+ e Modula-3, com base em ambientes mais poderosos.

Dentre estes ambientes, destacamos aqueles apoiados no sistema operacional UNIX em computadores VAX com processadores Motorola da serie 68000; no sistema operacional OS/2 em computadores PC com processadores Intel 80386/80486 com coprocessadores numéricos , ou, finalmente, em estações Modula-2 como o computador Lilith, que constitui uma máquina de alta desempenho, especificamente projetada para a execução de programas em Modula-2;

2 - Restrições ao conhecimento intensivo utilizado pelo sistema. Esta restrição, abre a possibilidade de se ampliar a base de conhecimento do sistema, visando a incorporação, sobretudo, de novas heurísticas. Esta extensão da base de conhecimento, deve ser baseada, em um trabalho contínuo, de análise e estudo de casos reais com o auxílio de operações simuladas. Neste sentido, reconhecemos que nossa base de conhecimento, constitui, apenas, um protótipo inicial, apresentando-se como ponto de partida para a elaboração de um modelo completo.

Este mesmo raciocínio, deve ser voltado para o desenvolvimento de novas medidas de reparo e ações de replanejamento, na fase de simulação de planos. Ademais, consideramos este processo de extensão e revisão de fundamental importância para a validação da base de conhecimento, bem como do sistema de planejamento como um todo;

3 - Restrições a manutenção de verdade e a existência de pré-condições necessárias ao exercício do planejamento. Neste caso, queremos relembrar que não consideramos na elaboração de planos e na simulação do planejamento, a ocorrência de diversas fontes de incerteza, sobretudo, aquelas que provocam a ocorrência de transformações que afetam atributos e valores estáticos da base de conhecimento, determinado, possivelmente, a inviabilização de ações na execução do planejamento.

Em segundo lugar, abordaremos as sugestões em relação a implementação do sistema de planejamento em tempo real, que dependerá, obviamente do ambiente de programação, ou do modelo de implementação escolhido:

1 - Definição e desenvolvimento de um modelo compátivel com os equipamentos de transmissão e aquisição de dados existentes no centro de controle de tráfego ferroviário onde se deseja implementar o sistema. Tal projeto deve considerar as seguintes etapas: análise do fluxo de dados ou informações; definição dos componentes como processos, atividades, canais de comunicação e estruras de armazenamento de dados; definição de interfaces e de mecanismos de comunicação e sincronização, e, finalmente, estudo dos mecanismos de interrupções causadas pela ocorrência de eventos externos, incluindo a adoção de prioridades.

Uma vez desenvolvida a implementação em tempo real, achamos que, inicialmente, o modelo deve ser testado, quanto as funções de aquisição de dados, em um centro de controle simulado. Posteriormente, o sistema deve funcionar em um centro real de operações de forma paralela, até que sua viabilidade seja completamente aferida;

2 - Estudo da interface homem x máquina, ou seja, elaboração de um projeto de comunicação entre o operador, o sistema SIGT, e o console de operação de trens e painel sinótico existentes no centro de controle da ferrovia;

3 - Estudo da monitoração ou avaliação periódica de parâmetros do planejamento, como exemplo, o posicionamento dos trens, considerando as restrições existentes nos equipamentos de transmissão e aquisição de dados, como exemplo, falta de sensores apropriados, rastreamento discreto, falta de contato terra-trem , etc.

4 - Adaptação do sistema à capacidade de processamento sob ausência de informações, considerando, a ignorância em relação a alguns eventos externos;

5 - Estudo para a atualização da base de conhecimento em tempo real, considerando o efeito de interrupções na realização de inferências;

6 - Avaliação mais precisa do tempo de resposta do sistema, em situações de alta imprevisibilidade que, certamente, existirão em função de deficiências de comunicação, exigindo maior poder de reatividade;

7 - Capacitação do sistema para operar de forma contínua, considerando, possivelmente, o desenvolvimento de interligações do tipo "hot stand-by", ou seja, em caso de suspensão de um sistema, o sistema reserva, devidamente atualizado , assume o controle do gerenciamento.

Também, deve ser considerado um tratamento adequado para exceções que causem a interrupção do sistema, bem como um mecanismo eficiente de coleta de lixo com, consequente, reutilização de memória.

Finalmente, abordaremos as questões básicas que envolvem o desenvolvimento de um sistema maior, de gerenciamento distribuído, envolvendo, desta forma, a coordenação de vários sub-sistemas localizados:

1 - Definição e desenvolvimento de um modelo hierárquico de gerenciamento, contendo um sistema supervisor, que estabelecerá as estratégias básicas de gerenciamento, incluindo o projeto de comunicação com os sub-sistemas localizados, e o conhecimento organizacional do sistema;

2 - Estudo da interface de comunicação dos sub-sistemas

localizados, ou, em outra palavras, a integração de planos adjacentes. Matematicamente, um estudo pelos métodos tradicionais de decomposição se mostrou inviável, face as restrições estruturais do problema. Neste sentido, deve-se procurar desenvolver métodos de Inteligência Artificial que envolvam a realização de planejamento na forma cooperativa com a existência de multi-agentes.

Concluindo, achamos que o cumprimento de tais restrições, ou a realização dos estudos propostos, demandaria um esforço de nossa parte, incompatível com as finalidades básicas da proposta de estudo original, necessitando, na maioria das vêzes, de um trabalho de equipe, com conhecimento mais amplo e amparo material que, momentaneamente, fogem ao nosso alcance.

Sendo assim, esperamos a compreenção a respeito das limitações de nosso trabalho e aguardamos,com toda certeza, a colaboração de novas pessoas , para que o mesmo se torne embrião de novos estudos.

Referências Bibliográficas:

A.A. Markov [1961] " Theory of Algorithms." Israel Program for Scientific Translation, Jerusalem

A.N.S. Freeling [1980], " Fuzzy Sets and Decision Analysis." IEEE Transactions on Systems, Man and Cybernetics, 10, 7, 341-354.

A. Church [1936] " An Unsolvable Problem of Elementary Number Theory." American Journal of Math. 58, 345-363

A. Kusiak e M. Chen [1988] " Expert Systems for Planning and Scheduling Manufacturing Systems." European Journal of Operational Research 34, 113-130

A.K. Mackworth e W.S. Havens [1981] " Structuring Domain Knowledge for Visual Perception." Proc. of the 7th Int. Joint Conf. on Artificial Intelligence, Vancouver, Canadian

A. Haas [1982] " Planning Mental Actions." TR106, Computer Science Dept., Univ. of Rochester, Rochester, NY

A. Haas [1985] " Possible Events, Actual Events and Robots." Comput. Intell. 1, 59-70

A. Monfroglio [1988] " Timetabling trhough a Deductive Database: A Case Study ." Data & Knowledge Engineering 3, 1-27

A.P. Dempster [1967] " Upper and Lower Probabilities Induced by a Multi-Valued Mapping." In the Annals of Mathematical Sta tistics 35-2, 325-339

A. Tate [1977] " Generating Project Networks ." In Proc. of the 5th Int. Joint Conf. on Artificial Intelligence, 888-893

A. Newell e H.A. Simon [1963] " GPS: A Program that Simulates Human Thought." In Computer and Thought , E.A. Feigenbaum e J. Feldman (Eds.), McGraw-Hill, NY

A. Newell [1980] " Reasoning, Problem Solving and Decision Processs: The Problem Space as a Fundamental Category. " Attention and Performance VIII, R. Nickerson (editor), Erlbaun, Hillsdale, New Jersey

A. Newell e H.A. Simon [1972] " Human Problem Solving." Prentice-Hall

A. Newell e H.A. Simon [1976] " Computer Sciences as Empirical Inquiry: Symbols and Search." Communications of the ACM 19(3), 113-126

A. Newell [1982] " The Knowledge Level." Artificial Intelligence 18, 87-127

A.A. Assad [1980] " Models for Rail Transportation." Transpn. Res. A, 14A, 205-220

A.Barr, P.R. Cohen e E.A. Feigenbaum [1981],[1982] " The Handbook of Artificial Intelligence, vols 1,2." Willian Kaufmann Inc.

A.P. Sage e C.C. White [1984] " ARIADNE: A Knowledge-Based Interactive System for Planning and Decision Support." IEEE transactions on Systems, Man and Cybernetics, 14(1), 35-47

A.T. Bahill, P. Harris e E. Senn [1985] " Lessons Learned Building Expert Systems." AI Expert September, 36-45

B.G. Buchanan e E.H. Shortliffe [1984] " Rule-Based Expert Systems: The MYCIN Experiment of the Stanford Heuristic Programming Project." Addison Wesley Publ. Co.

B. Liskov e J. Guttag [1986] " Abstraction and Specification in Program Development " The MIT Press, Cambridge, Massachusetts

B. Liskov [1987] " Data Abstraction and Hierarchy." in OOPSLA 87 Addendum to the Proc., SIGPLAN Notices October, 17-34

B. Stroustrup [1986] " The C++ Programming Language." Addison Wesley Publ. Co.

B. Szpigel [1972] " Optimal Train Scheduling on a Single Track Railway." OR'72, North Holland, Amsterdam, 345-352

B. Werners [1987], " Interactive Multiple Objective Programming Subject to Flexible Constraints." European Journal of Operational Research, 31, 342-349.

B. Hayes-Roth [1985] " A Blackboard Architecture for Control." Artificial Intelligence 26, 251-321

B. Hayes-Roth e F. Hayes-Roth [1979] " A Cognitive Model of Planning." Cognitive Science 3, 275-310

C.E. Bell [1989] " Reasoning with Disjunctive Temporal Constraints." Paper, Department of Management Sciences, University of Iowa, Iowa city, CA

C.P. Langlotz e E.H. Shortliffe [1989] " Logical and Decision-Theoretic Methods for Planning Under Uncertainty." AI Magazine Spring, 39-47

C.A. O'Reilly e A.S. Cromarty [1985] " 'Fast' is not 'Real-Time' in Designing Effective Real-Time AI Systems." in Application of AI II 548, Bellingham, 249-257

C.F. Wagner [1988] " Implementing Abstraction Hierarchies." Trabalho Técnico, Departamento de Ciência de Computação, UNICAMP, Campinas

C.J. Hogger [1984] " Introduction to Logic Programming." Academic Press Inc.

C.L. Forgy e J. McDermott [1976] " The OPS reference manual." Techinical Report, Department of Computer Science, Carnegie-Mellon University

C.L. Forgy [1982] " Rete: A fast Algorithm for the Many Pattern / Many Object Match Problem." Artificial Intelligence 19, 17-37

Canadian National Rail [1981] " Route Capacity Model (RCM)." Company Report

C. J. Date [1986] " Introdução a Sistemas de Banco de Dados." Editora Campus, Rio de Janeiro

C.V. Negoita [1981] " The Current Interest in Fuzzy Optimization." Fuzzy Sets and Systems 6, 261-269

C.V. Negoita e D.A. Ralescu [1975], " Applications of Fuzzy Sets and Systems Analysis. " Birkhauser, Basel, 12-24.

C.V. Negoita [1985], " Expert Systems and Fuzzy Systems." The Benjamin/Cummings Publishing Company, Inc..

D.A. Sewry [1984a] " Modula-2 Process Facilities." SIGPLAN Notices 19, 11, 23-32

D.A. Sewry [1984b] " Modula-2 and the Monitor Concept. " SIGPLAN Notices 19, 11, 33-41

D. Dubois e H. Prade [1980], " Systems of Linear Fuzzy Constraints." Fuzzy Sets and Systems 3, 37-48.

D. Ben-Arieh [1986] " Knowledge-based Control System for Automated Production and Assembly." in Modelling and Design Flexible Manufacturing Systems, A. Kusiak ed. , Elsevier, New York, 347-368

D.G. Bobrow [1985] " IF Prolog is the Answer, What is the Question? or What it Taks to Support AI Programming Paradigms." IEEE Transactions on Software Engineering 11(11), 1401-1408

D.G. Bobrow e T. Winograd [1977] " An Overview of KRL: A Knowledge Representation Language." Cognitive Science 1, 3-46

D.L. Parnas [1972] " On the Criteria to be Used in Composing Systems into Modules." Communications of the ACM 15(12)

D. R. Edelson [1987] " How Objective Mechanisms Facilitate the Development of Large Software Systems in three Programming Languages." SIGPLAN Notices 22(9), 54-63

D. Rosenthal [1985] " Adding Meta Rules to OPS5: A Propostal Extension." SIGPLAN Notices 20(10), 79-86

D. Kumar [1986] " A Novel Approach to Sequential Simulation." IEEE Software, September

D.N. Chorafas [1987] " Applying Expert Systems in Business." McGraw-Hill Book Co.

D. Budgen [1985] " Combining MASCOT with Modula-2 to Aid the Engineering of RealTime Systems." Software Practice and Experience 15(8), 767-793

D. Dubois e H. Prade [1980] " Systems of Linear Fuzzy Constraints." Fuzzy Sets and Systems 3, 37-48

D.E. Wilkins [1980] " Using Patterns and Plans in Chess. " Artificial Intelligence 14, 165-203

D.E. Wilkins [1984] " Domain Independent Planning : Representations and Plan Generation."Artificial Intelligence 22, 269-301

D.E. Wilkins [1985] " Recovering from Execution Errors in SIPE." Comput. Intell. 1, 33-45

D. Chapman [1987] " Planning for Conjuctive Goals ." Artificial Intelligence 32, 333-377

D. McDermott [1988] " Planning and Execution." In Report of DARPA Workshop on Planning, W.Swartout Ed., AI Magazine Summer, 115-131

D. McDermott [1982] " A Temporal Logic for Reasoning about Processes and Plans." Cognitive Science 6, 101-155

D. McDermott e J. Doyle [1980] " Non-Monotonic Logic I." Artificial Intelligence 13, 41-72

D.J.H. Brown e S. Dowsay [1979] " The Challeng of GO." New Scientist, 81, 303-305

E.P.D. Pednault [1986], " Toward a Mathematical Theory of Plan Synthesis." PHD Thesis, Dep. of Electrical Engineering, Stanford University, Stanford.

E.L. Hannan [1981], " Linear Programming with Multiple Fuzzy Goals." Fuzzy Sets and Systems 6, 235-248.

E.A. Feigenbaum e J. Feldman (editors) [1963] " Computers and Thought." MacGraw-Hill Inc.

E. Charniak e D. McDermott [1985] " Introduction to Artificial Intelligence." Addison-Wesley Publ. Co.

E. Balas [1969] " Machine-Sequening via Disjunctive Graphs: An Implicit Enumeration Algorithm." Operations Research 17, 941-957

E.A. Rich [1986] " Rationality and Creativity in Artificial Intelligence. " Expert Systems and Knowledge Engineering, T. Bernold Ed., Elsevier, North Holland , 7-16

E.A. Rich [1983], " Artificial Intelligence." McGraw-Hill Co..

E.D. Sacerdoti [1974] " Planning in a Hierarchy of Abstraction Spaces." Artificial Intelligence 5, 115-135

E.D. Sacerdoti [1977] " A Structure for Plans and Behavior." American Elsevier Publs. Co.

E.D. Sacerdoti [1975] " The Nonlinear Nature of Plans." In Proc. of the 4th Int. Joint Conf. on Artificial Intelligence

E.P.K. Tsang [1988] " Time Structures for AI." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 456-461

E. L. Post [1936] " Finite Combinatory Processes-Formulation, I." Symb. Logic 1, 103-105

E.L. Post [1943] " Formal Reductions of the General Combinatorial Problem." Amer. J. Math. 65

E. Nuutila, J. Kuusela, M. Tamminen, J. Keilahti, J. Arkko e N. Bouteldja [1987] " XC - A Language for Embedded Rule-Based Systems." SIGPLAN Notices 22(9), 23-32

F.R.D. Velasco [1987] " Uma Implementação da linguagem OPS5 para Computadores Compatíveis com IBM-PC." 4o SBIA, Uberlândia

F.C. Dargam, E.P. Passos e F.R. Pantoja [1988] " Desenvolvimento do Sistema SEGE - S.E. Aplicado a Guerra Eletronica." 5o SBIA, Natal

F. Hayes-Roth, D.A. Waterman e D.B. Lenat [1983] " Building Expert Systems." Addison-Wesley Publ. Co.

G.A. Ford e R.S. Wiener [1985] " Modula-2 : A Software Development Approach." John Wiley & Sons

G. Barney [1986] " Intelligent Instrumentation." Prentice-Hall

G.W. Cherry [1984] " Parallel Programming in Ansi Standard ADA." Prentice-Hall

G. Chryssolouris [1989] " On Intelligent Manufacturing Systems. "12th IMACS (World Congress on Scientific Computation), Paris,july,18-22

G.A. Gorry e M.S. Norton [1971] " A Framework for Management Information Systems." Sloan Management Review, 13(1), 55-70

G.P. Zarry [1986] " Constructing and Utilizing Large Fact Databases using Artificial Intelligence Techniques." in Proceedings of the First International Workshop on Expert Database System, Larry Kerschberg (editor), Benjamin/Cumnings Publ. Co.

G. Cohen, D. Dubois, J-P Quadrat e M. Viot [1985] " A Linear-System- Theoretic View of Discrete-Event Processes and its use for Performance Evaluation in Manufacturing." IEEE transactions on Automatic Control, 30(3), 210-220

G. Brewka [1987] " The Logic of Inheritance in Frame Systems." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence,483-488

G. Bruno, A. Elia e P. Laface [1986] " A Rule-Based System to Schedule Production." IEEE Computer, july, 32-40

G. Bruno [1984] " Rationale for the Introduction of Discret Simulation Primitives in ADA." in Proc. of the Conference on Simulation in Strongly Typed Languages, R. Bryant e B.W. Unger Eds. San Diego,10-15

G.M. Birtwistle, O-J. Dahl, B, Myhrhang e K. Nyguard [1973] " SIMULA Begin." Auerbach

G.T. Vesonder [1988] " Rule Base-Programming in the UNIX System." AT&T Technical Journal 67,1, 69-80

G.J. Stoetel, R.D. Baxter e M.J. Denney [1986] " A Capacity Planning Expert Systems for IBM System/38." IEEE Computer July , 42-50

G.J. Sussman [1975] " A Computer Model of Skill Acquisition." American Elsevier Publs. Co.

G. Shafer [1976] " A Mathematical Theory of Evidence." Princeton University Press

H.A. Simon [1973] " The Structure of Ill-Structured Problems." Artificial Intelligence 4, 181-200

H. A. Simon [1987] " Two Heads Are Better Then One : The Collaboration Beteween AI and OR." Interfaces 17(4), 8-15

H.A. Simon [1981] " The Sciences of Artificial." The Massachusetts Institute of Technology Press

H.A. Simon [1983], " Search and Reasoning in Problem Solving." Artificial Intelligence 21, 7-29

H.A. Simon [1986], " Whether Software Engineering Needs to be Artificial Intelligent. " IEEE Transactions on Software Engineering, 12, 7, 726-732.

H.H. Greenberg [1968], " A Branch and Bound Solution to the General Scheduling Problem." Operations Research, 16, 353-361.

H.A. Kautz e E.P.D. Pednault [1988], " Planning and Plan Recognition." AT&T Techinical Journal, 67, 1, 25-39.

H. Prade [1979], " Using Fuzzy Set Theory in a Scheduling Problem. " Fuzzy Sets and Systems 2, 153-165.

H. Tanaka e K. Asai [1984], " Fuzzy Linear Programming Problems with Fuzzy Numbers." Fuzzy Sets and Systems 13, 1-10.

H.-J. Zimmermann [1978], " Fuzzy Programming and Linear Programming with Several Objective Functions." Fuzzy Sets and Systems 1, 45-55.

H.-J. Zimmermann e P. Zysno [1980], " Latent Connectives in Human Decision Making." Fuzzy Sets and Systems, 4, 37-51.

H.-J. Zimmermann [1988] " Modelling, Flexibility, Vagueness and Uncertainty in Operations Research." Investigacion Operativa 11, 7-34

H. Levesque e J. Mylopoulos [1979] " A Procedural Semantics for Semantic Networks in Associative Networks." N. Findler Ed., Academic Press

H.P. Nii [1987] " Blackboard Systems." AI magazine Spring , 38-53 e Summer , 82-106

J. G. Vaucher e P. Durval [1975] " A Comparison of Simulation Event List Algorithms." Communications of the ACM 10,4, 223-230

J. Hooger [1984] " Introduction to Logic Programming. " Academic Press Inc.

J. Bergin e S. Greenfield [1987] " What Does Modula-2 Need to Fully Suport Object Oriented Programming." Division of Computer Science and Mathematics, Marist College, New York

J.A. Robinson [1983] " Logic Programming - Past, present and future." New Generation Comput. 1, 2

J.C. Ferguson, R. Siemens e R.E. Wagner [1985] " Star-Plan : A satellite anomaly resolution and planning system." Intern. Ford Aeroespace and Communications Corp. , California

J. Doyle [1985] " Reasoned Assumptions and Pareto Optimality." In Proc. of the 9th Int. Joint Conf. on Artificial Intelligence, Menlo Park, CA, 88-90

J. Doyle [1979] " A Truth Maintenance System." Artificial Intelligence 12, 231-272

J.F. Allen [1984] " Towards a General Theory of Action and Time." Artificial Intelligence 23, 123-154

J.F. Allen e P.J. Hayes [1985] "A Common-sense Theory of Time." In Proc. of the 9th Int. Joint Conf. on Artificial Intelligence, Menlo Park, CA

J.F. Allen [1983] " Maintaining Knowledge about Temporal Intervals." Communications of the ACM 26,11, 832-843

J.J. Baldwin [1979] " A New Approach to Approximate Reasoning Using a Fuzzy Logic." Fuzzy Sets and Systems 2, 309-325

J.P. Stinson, E.W. Davis e B.M. Khumawala [1978] " Multiple Resource-Constrained Project Scheduling Using Branch and Bound." AIIE transactions 10,3, 252-259

J. McCarthy e P.J. Hayes [1969] " Some Philosophical Problems from Standpoint of Artificial Intelligence, Machine Intelligence 4, 463-502

J. Guttag [1977] , " Abstract Data Types and the Development of Data Structures." Communications of the ACM 20,6, 396-404

J.E. Laird, A. Newell e P.S. Rosenbloom [1987] " SOAR: An Architecture for General Intelligence." Artificial Intelligence 33, 1-64

J. McDermott [1982] " R1: A Rule-Based Configurer of Computer Systems." Artificial Intelligence 19, 39-88

J. Robertson [1985] " 'STIMULUS' - A Base Language for Real Time Expert Systems." in Proc. Conf. on AI and advanced Computer Technology, Wiesbaden, 24-26

J. Savit [1987] " Uninitialized Modula-2 Abstract Objects, revisited." SIGPLAN Notices 22, 2, 78-84

J.H. Moore e M.G. Chang [1980] " Design of Decision Support Systems." Data Base 12 1(2), 8-14

J. Pearl [1984] " Heuristics: Intelligent Search Strategies for Computer Problem Solving." Addison-Wesley Publ. Co.

J.J. Buckley [1983] " Fuzzy Programming and the Pareto Optimal Set." Fuzzy Sets and Systems, 10, 57-63

J.D.C. Little [1970] " Models and Managers: The Concept of a Decision Calculus." Management Science 16(18), B466 - B485

J-P. Briot e P. Cointe [1987] " A Uniform Model for Object-Oriented Languages Using the Class Abstraction." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 40-42

J.L. Verdegay [1984], " A Dual Approach to Solve the Fuzzy Linear Programming Problem." Fuzzy Sets and Systems 14, 131-141.

J.A. Goguen [1967], " L-Fuzzy Sets." J. Math. Anal. Appl. 18, 145-174.

J.M. Charlton e C.C. Death [1970], " A Generalized Machine Scheduling Algorithm." Operational Research Quarterly, 21, 127-134.

K.J. Healey [1986] " Artificial Intelligence Research and Applications at the NASA Johnson Space Center." AI Magazine 7(3), 146-152

K.J. Fordyce e G.A. Sullivan [1986] " Decision Simulation (DSIM) - One Outcome of Combining Expert Systems and Decision Support Systems." in Artificial Intelligence in Economics and Management, L.F. Pau (editor), North-Holland Publ. Co., 31-40

K. Kant [1981], " Dynamic Railway Traffic Control: A New Approach. " Rail International, june , 257-264.

L. Lamport [1978] " Time, Clocks, and the Ordering of Events in a Distributed System." Communications of the ACM, 21,7, 558-565

Logitech Modula-2/86 [1986] " User's Manual Release 2.0", Logitech Inc.

L.F.B. Baptistella e A. Ollero [1980] " Fuzzy Methodologies for Interactive Multicriteria Optimization. " IEEE Transactions on Systems, Man and Cybernetics, 10, 7, 355-365

L.A. Zadeh [1978], " Fuzzy Sets as a Basis for a Theory of Possibility. " Fuzzy Sets and Systems 1, 3-28.

L.A. Zadeh [1983], " The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems." Fuzzy Sets and Systems, 11, 199-226.

L.A. Zadeh [1975] " Fuzzy Logic and Aproximate Reasoning." Synthese 30, 407-428

L.A. Zadeh [1983b] " Commo-sense Knowledge Representation Based on Fuzzy Logic." IEEE Computer October, 61-65

L.D. Erman, F. Hayes-Roth, V.R. Lesser e D.R. Reddy [1980] " The Hersay II Speech-Understanding System: Integration Knowledge to Resolve Uncertainty." Computing Surveys 12, 213-253

L. Morgenstern [1988] " Knowledge Preconditions for Actions and Plans." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 867-874

M. Minsky [1975] " A Framework for Representing Knowledge." in The Psychology of Computer Vision, P. Winston Ed., New York, 119-185

M.P. Georgeff [1982] " Procedural Control in Production Systems." Artificial Intelligence 18, 175-201

M.H. MacDougall e J.S. MacAlpine [1973] " Computer System Simulation with ASPOL." in Proc. of the Symposium on the Simulation of Computer Systems , June, 92-103

M.J. Schoppers [1988] " Universal Plans for Reactive Robots in Unpredictable Environments." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 1039-1046

M. Mizumoto e H.-J. Zimermann [1982] " Comparison of Fuzzy Reasoning Methods." Fuzzy Sets and Systems 8, 253-283

M.P. Wellman [1988] " Dominance and Sub-Assumption in Constraint-Posting Planning." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 884-890

M.J. Stefik [1981a] " Planning with Constraints (MOLGEN: Part 1)." Artificial Intelligence 16, 111-140

M.J. Stefik [1981b] " Planning and Meta-Planning (MOLGEN: Part 2)." Artificial Intelligence 16, 141-170

M.J. Stefik [1979] " An examination of a frame-structured representation system." in Proc. of the 6th Int. Joint Conf. on Artificial Intelligence, Tokyo, August, Kaufmann, Los Altos, California, 845-852

M.J. Stefik, D.G. Brobow, S. Mittal e L. Conway [1983] " Knowledge Programming in LOOPS: report of a experimental course." AI Magazine Fall

M.J.P. Shaw e A.B. Winston [1986] " Application of Artificial Intelligence to Planning and Schedulling in Flexible Manufacturing." in Flexible Manufacturing Systems: Methods and Studies, A. Kusiak ed., North-Holland, Amsterdam, 223-242

M.J.P. Shaw e A.B. Winston [1985] " Task Bidding and Distributed Planning in Flexible Manufacturing." The second Conference on Artificial Intelligence Applications, Miami Beach, Florida, December, 184-189

M.M. Syslo , N. Deo e S. Kowalik [1983] " Discret Optimization Algorithms." Prentice-Hall

M.S. Fox [1983] " The Intelligent Management System: An

Overview." in Processes and Tools for Decision Support, H.G. Sol (editor), North-Holland Publ. Co., 105-130

M.S. Fox e J. McDermott [1986] " The Role of Databases in Knowledge -Based System. " On Knowledge Base Management Systems, Michael Brodie and John Mylopaulos (editors), Springer-Verlag, 409-430

M.S. Fox [1989], " Reflections on the Relationship Between Artificial Intelligence and Operations Research ."IN Proceedings of The NASA Conference on Space Telerobotics, Pasadena, CA, USA

M.J. Ginzberg e E.A. Stohr [1982] " Decision Support Systems: Issues and Perspectives." in Decision Support Systems, M.J. Ginzberg, W. Reitman e E. A.Stohr (editors), North-Holland Publ. Co.

M.K. Luhandjula [1982], " Compensatory Operators in Fuzzy Linear Programming With Multiple Objectives. " Fuzzy Sets and Systems 8, 245-252.

M.J. Folk e B. Zoellick [1987] " Files Structures: A Conceptual Toolkit. " Addison-Wesley Publ. Co.

M. Sakawa e H. Yano [1986], " Interactive Fuzzy Decision Making for Multiobjective Nonlinear Programming Using Augmented MinMax Problems." Fuzzy Sets and Systems 20, 31-43.

N. Maculan Filho [1978], " Programação Linear Inteira." COPPE, Comissão de Publicações, PDD 17/78.

N.J. Nilsson [1980], " Principles of Artificial Intelligence." Tioga Publishing Co..

N.C. Rowe [1988] " Artificial Intelligence trough Prolog." Prentice-Hall Int. Inc.

N. Wirth [1988] " From Modula to Oberon." Software - Practice and Experience 18(7), 661-670

N. Wirth [1988] " Type Extensions." ACM Transactions on Programming Language and Systems 10(2), 204-214

N. Wirth [1982] " Programming in Modula-2." Heidelberg: Springer-Verlag

N.M. Morris e W.B. Rouse [1985] " The Effects of Type of Knowledge upon Human Problem Solving in a Process Control Task ." IEEE transactions on Systems, Man and Cybernetics, 15(6), 698-707

O.J. Dahl e K. Nygaard [1966], " Simula - an Algol-Based Simulation Language." Communications of the ACM, 9,9, 671-678

P. B. Hansen [1972], " Structured Multi-Programming." Communications of the ACM, 15,7, 574-578

P. B. Hansen [1973] " Operation Systems Principles ." Englewood Cliffs, NJ, Prentice-Hall

P.A. Rubin e R. Narasimhan [1984], " Fuzzy Goal Programming

with Nested Priorities. " Fuzzy Sets and Systems 14, 115-129.

P. Robinson e M. Jordan [1988] " A Programming Environment for Modula-2." Software Engineering Journal, july, 119-126

P.E. Hart, N.J. Nilsson e B. Raphael [1968], " A Formal Basis for the Heuristic Determination of Minimum Cost Paths. " IEEE Transactions of Systems, Science and Cybernetics, 4, 2, 100-107.

P.S. Rosenbloom e A. Newell [1986] " The Chunking of Goal Hierarchies: A Generalized Model of Practice." In R.S. Michalski, J.G. Carbonell e T.M. Mitchell, eds., Machine Learning: An Artificial Intelligence Approach, vol. 2, Morgan-Kaufmann.

P. Rovener [1986] " On extending Modula-2 for building large, integrated systems." IEEE Software, 3(6), 18-26

P.A. Newman e K.G. Kempf [1985] " Opportunistic Scheduling for Robotic Machine Tending". The second Conference on Artificial Intelligence Applications, Miami Beach, Florida, December, 168-173

P.A. Subramanyam [1985] " The Software Engineering of Expert Systems: Is Prolog Appropriate ?" IEEE Transactions om Software Engineering SE-11, 11, 1391-1400

P.A. Subramanyam e R.G. Askin [1986] " An Expert System Approach to Scheduling in Flexible Manufacturing Systems." in Flexible Manufacturing Systems: Methods and Studies, A. Kusiak ed., North-Holland, Amsterdam, 243-256

P. Szolovits [1986] " Knowledge-Based Systems: A Survey." On Knowledge-Based Management Systems, M. Brodie e J. Mylopoulos editors, Springer-Verlag, 339-352

P. Wegner [1986] " Classification in Object-Oriented Systems." SIGPLAN Notices , 21, 10. 173-182

Peat, Marwick, Mitchell and Co. [1975] " Train Dispatching and Simulation Model: Capabilities and Description." Company Report

P. Pin-Sham Chen [1976] " The Entity-Relationship Model - Toward a Unified View of Data." ACM Transactions on Database Systems 1(1), 9-36

P.G.W. Keen [1980] " Adaptive Design for Decision Support Systems." Database, 12 1(2)

P.H. Bonczeck, C.W. Holsapple e A.B. Whinston [1980] " The Evolving Roles of Models in Decision Support Systems." Decision Sciences, 11(2), 339-356

P.E. Lehner [1986] " On the Role of Artificial Intelligence in Command and Control." IEEE transactions on Systems, Man and Cybernetics , 16(6), 824-833

P.R. Young e P.E. Lehner [1986] " Application of a Theory of Automated Adversarial Planning to Command and Control." IEEE transactions on Systems, Man and Cybernetics 16, 6, 806-812

P.S. Rosenbloom, J.E. Laird, J. Mcdermott, A. Newell e E. Orciuch [1985] " R1-SOAR: An Experiment in Knowledge-Intensive Programming in a Problem-Solving Architecture." IEEE Transactions on PAMI 7,5, 561-569

P. Friel e S. Sheppard [1985] " Implications of the ADA Environment for Simulation Studies." Simuletter, 16,2, 14-26

P. Harmon e D. King [1985] " Artificial Intelligence in Bussiness." John Wiley & Sons

P. Klahr [1984] " Artificial Intelligence Approaches to Simulation." in Proceedings of the UKSC Conference on Computer Simulation, London, England, 87-92

P. Cheeseman [1988] " Uncertainty and Planning: A Summary." In Report of DARPA Workshop on Planning , W. Swartout Ed. , AI Magazine Summer , 115-131

P.J. Hayes [1973] " The Frame Problem and Related Problems in AI." In Artificial and Human Thinking, A. Elithorn e D. Sorer (Eds.), San Francisco, CA

P.J. Hayes e J.F. Allen [1988] " Short Time Periods." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 981-983

P.R. Bonissone e A.L. Brown Jr. [1986] " Expanding the Horizons of Expert Systems." In Expert Systems and Knowledge Engineering, T. Bernold (Ed.), 267-288

R. Sharma e L.L. Rose [1988] " Modular Design for Simulation." Software-Practice and Experience 18,10, 945-966

R. Ohran [1984] " Lilith and Modula-2." Byte, august, 181-192

R.E. Bellman, A.O. Esogbue e I. Nabeshina [1982], " Mathematical Aspects of Scheduling and Applications." Pergamon Press.

R.E. Bellman e L.A. Zadeh [1970], " Decision-Making in a Fuzzy Environment." Management Sci. 17, 141-164.

R.E. Bellman e L.A. Zadeh [1976], " Local and Fuzzy Logics.", Electronics Research Laboratory, College of Engineering, Univ. California, Berkeley, Tech. Report ERL M584.

R. Detcher e J. Pearl [1985], " Generalized Best-First Search Strategies and the Optimality of A*." Journal of the ACM, 32, 3, 505-536.

R. Jain [1976], " Decisionmaking in the Presence of Fuzzy Variables." IEEE Transactions on Systems, Man and Cybernetics, october 1976, correpondence, 698-703.

R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan [1977], " Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. " Discrete Optimization 77 Seminar, Vancuver, Canada.

R.E. Campelo e N. Maculan Filho [1989] " Projeto e Análise de Algoritmos Heurísticos." Textos da I Escola Brasileira de Otimização, Rio de Janeiro

R.E. Korf [1985] " Deph-First Iterative-Deepening: An Optimal Admissible Tree Search." Artificial Intelligence 27, 97-109

R.E. Korf [1987] " Planning as Search: A Quantitative Approach. " Artificial Intelligence, 33, 65-88

R.E. Korf [1985] " Macro-Operators : A Weak Method of Learning." Artificial Intelligence, 26, 35-77

R.M. Bryant [1982] " Discret System Simulation in ADA." Simulation, October, 111-121

R. Slowinski [1986] " A Multi-criteria Fuzzy Linear Programming Method for Water Supply System Development Planning." Fuzzy Sets and Systems, 19, 217-237

R.M. Kerr e R.V. Ebsary [1988] " Implementation of an Expert System for Production Scheduling." European Journal of Operational Research 33, 17-29

R.J. Brachman e J.G. Schmolze [1985] " An Overview of the KL-ONE Knowledge Representation System." Cognitive Science 9, 2 , 171-216

R.J. Brachman [1988] " The Basics of Knowledge Representation and Reasoning." AT&T Technical Journal 67,1, 7-24

R. Terada [1982] " Desenvolvimento de Algoritmos e Complexidade de Computação." III Escola de Computação, PUC/RJ, Rio de Janeiro

R.S. Wiener [1986] " Protecting Against Uninitialized Abstract Objects in Modula-2." SIGPLAN Notices 21, 6, 63-69

R.D.P. Barbosa e outros [1986] " Controle de Tráfego Centralizado Utilizando Controladores Programáveis." Revista Ferroviária, Setembro, 25-28

R. Reddy [1988] " Foundations and Grand Challengs of Artificial Intelligence." AI Magazine, winter, 9-21

R. King [1986] " A Database Management System Based on an Object Oriented Model." Proceedings of the First International Workshop on Expert Database System, Larry Kerschberg (editor), Benjamin/Cumnings Publ. Co.

R. Fonseca Neto e A.A. Pinho [1988] "Utilização de Algoritmos de Busca Informada em um Problema de Programação Discreta com Aplicação na Otimização da Composição de Locomotivas em Trens Unitários." Resumos do IV Congresso Latino-Íbero-Americano de Pesquisa Operacional e Engenharia de Sistemas, outubro, Rio de Janeiro

R. Fonseca Neto [1988] " Flow-Shop Scheduling with Fuzzy Numbers ." Relatório Interno, COPPE / Dept. de Engenharia de Sistemas e

Computaçãо, Rio de Janeiro

R. Davis [1980] " Reasoning about Control. " Artificial Intelligence 15, 179-222

R.E. Fikes e T. Kehler [1985] " The Role of Frame-Based Representation on Reasoning." Communications of ACM, 28(9), 904-920

R.E. Fikes e N.J. Nilsson [1971] " STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." Artificial Intelligence 2, 189-208

R.E. Fikes [1988] " The Use of Truth Maintenance in Planning Systems." In Report of DARPA Workshop on Planning , W. Swartout Ed. , AI Magazine Summer , 115-131

R.E. Fikes, P.E. Hart e N.S. Nilsson [1972] " Learning and Executing Generalized Robot Plans." Artificial Intelligence 3, 251-288

R. Dodhiawala, V. Jagannathan, C. Baum e T. Skilman [1989] " The First Workshop on Blackboard Systems." AI Magazine Spring, 77-80

R. Wilensky [1983] " Planning and Understanding." Addison-Wesley Pubs. Co.

R.C. Moore [1980] " Reasoning about Knowledge and Action." TR 191, SRI, AI Center, Menlo Park, CA

R.P. Bossano [1989] " What AI Can Do for Battle Management." Workshop Report, AI Magazine, Fall, 77-83

R.M. Stallman e G.J. Sussman [1977] " Foward Reasoning and Dependency-Direct Backtracking in a System for Computer-Aided Circuit Analysis", Artificial Intelligence 9, 135-196

R. Reiter [1980] " A Logic for Default Reasoning." Artificial Intelligence 13, 81-132

R.E. Shannon, R. Mayer e H.H. Adelsberg [1985] " Expert Systems and Simulation." Simulation, 44(6), 275-284

R. O'Keefe [1986] " Simulation and Expert Systems: A Taxonomy and Some Examples." Simulation, 46(1), 10-16

R. Dreyfus [1989] "Expert Systems: How Far Can They Go?." In Panel session at the 9th Int. Joint Conf. on Artificial Intelligence, Menlo Park, CA, R. Davis (Ed.), Part One, AI Magazine, Spring, 61-67

S.C. Graves [1981], " A Review of Production Schedulling ". Operations Research, 29, 4, 646-675.

S.E. Cross [1985], " Computer Understanding of Air Traffic Control Displays. " IEEE Transactions on Systems, Man and Cybernetics, 15, 1, 133-135.

S.L. Alter [1980] " Decision Support Systems: Current Practices and Continuing Challenges, Reading, Massachusetts, Addison-Wesley Publ. Co.

Se-Young Oh [1986] " A Pattern Recognition and Associative Memory Approach to Power System Security Assessment." IEEE transactions on Systems, Man and Cybernetics, 16(1), 62-72

S. Bennett [1988] " Real-Time Computer Control: An Introduction." Prentice-Hall

S. Greenfield e R. Norton [1987] " Detecting Uninitialized Modula-2 Abstract Objects." SIGPLAN Notices 22, 6, 52-58

S. Sheppard, P. Friel e D. Reese [1985] " Simulation in ADA: an implementation of two world views." in Proc. of the Conference on Simulation in Strongly Typed Languages, R. Bryant e B.W. Unger Eds. San Diego, CA, 3-9

S. Shen e Y. Chang [1986] " An AI Approach to Schedule Generation in a Flexible Manufacturing System." in Flexible Manufacturing Systems : Operations Research Models and Applications, K.E. Steck e R. Suri eds., Elsevier, New York, 581-592

S.A. Vere [1983] " Planning in Time: Windows and Durations for Activities and Goals." IEEE transactions on PAMI 5,3, 246-267

T.E. Morton e T.L. Smunt [1986] " A Planning and Scheduling System for Flexible Manufacturing." in Flexible Manufacturing Systems: Method and Studies, A. Kusiak ed., North Holland, Amsterdam, 151-164

T.J. Laffey, P.A. Cox, J.L. Schmidt, S.M. Kao e J.Y. Reed [1988] " Realtime Knowledge-Based Systems." AI Magazine Spring, 27-45

T.P. Kehler e G.D. Clemenson [1984] " An Application Development System for Expert Systems." System Software 3(1), 212-224

T. Xu-Yen [1986] " Intelligent Control and Intelligent Management for Large Scale Systems." in Artificial Intelligence in Economics and Management, L.F. Pau (editor), North-Holland, 87-92

T.S. Larkin, R.I. Carruthers e R.S. Soper [1988] " Simulation and Object-Oriented Programming: The Development of SERB." Simulation, 52(3), 93-101

T.A. Linden e J. Glicksman [1988] " Contigence Planning for an Autonomous Land Vehicle." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 1047-1054

T.A. Linden [1988] " Plan and Goal Representations." In Report of DARPA Workshop on Planning, W.Swartout Ed.,AI Magazine Summer,115-131

T.L. Dean e D.V. McDermott [1987] " Temporal Database Management." Artificial Intelligence 32

T.L.Dean [1988] " Planning Paradigms." In Report of DARPA Workshop on Planning , W. Swartout Ed. , AI Magazine Summer , 115-131

T.L. Dean [1988] " Large-Scale Temporal Data-base for Planning in Complex Domains ." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 860-866

T. Winograd [1989] "Expert Systems: How Far Can They Go?." In Panel session at the 9th Int. Joint Conf. on Artificial Intelligence, Menlo Park, CA, R. Davis (Ed.), Part One, AI Magazine, Spring, 61-67

U. Thole, H.-J.Zimmermann e P. Zysno [1979], " On the Suitability of Minimum and Product Operators for the Intersection of Fuzzy Sets." Fuzzy Sets and Systems, 2, 167-180.

V.R. Lesser, J. Pavlin e E. Durfee [1988] " Approximate Processing in Real-Time Problem Solving." AI Magazine Spring, 49-61

W.B. Rauch-Hindin [1985] " Artificial Intelligence in Business, Science and Industry, Vol 1,2." Prentice-Hall

W.S. Havens [1985] " A Theory of Schema Labelling." Comput. Intell. 1, 127-139

W. Reitman [1982] " Applying Artificial Intelligence to Decision Support: Where Good Alternatives Come From." in Decision Support Systems, M.J. Ginzberg, W. Reitman e E. A.Stohr (editors), North Holland Publ. Co.

W. Swartout [1988] " DARPA Worksshop on Planning." AI Magazine, summer,115-130

W.R. Franta [1977] " Process View of Simulation." Elsevier, North-Holland

W.R. Franta e K. Maly [1977] " An efficient Data Structure for the Simulation Event Set." Communications of the ACM, 20,8, 596-602

W. Kreutzer [1979]," Patterns of Modelling: Towards a Conceptual Basis for Discret Event Simulation." Simuletter, 7-22

Y. Lirov, E.Y. Rodin, B.G. McElhaney e L.W. Wilbur [1988] " Artificial Intelligence Modelling of Control Systems." Simulation, 50(1), 12-24

Ye-Sho Chen [1988] " Integrating Data, Model and Knowledge Management : An Entity-Relationship Approach." TIMS/ORSA Conference, Washington, USA, April, 25-27

Y.V. Reddy, M.S. Fox e N. Husain [1985] " Automating the Analysis of Simulation in KBS." in AI Graphics and Simulation, G. Birtwistle (editor), The Society of Computer Simulation, San Diego, USA, 34-40

Y. Kobayashi, Y. Wada e T. Kiguchi [1986], " Knowledge Representation and Utilization for Optimal Route Search." IEEE Transactions on Systems, Man and Cybernetics, 16, 3, 454-462.

Y. Shoham [1988] " Non-Monotonic Logics: Meaning and Utility." In Proc. of the 10th Int. Joint Conf. on Artificial Intelligence, 388-393

Z. Manna [1974] " Mathematical Theory of Computation." MacGraw-Hill Inc.

Apêndice I - Esquema de Classificação do Sistema SIGT

Neste apêndice, apresentamos um resumo dos principais modelos e métodos empregados em nosso sistema , através de um esquema de classificação que procura especificar as características básicas da modelagem matemática e do modelo de programação "fuzzy" desenvolvido para o problema de sequenciamento de trens , do método de solução empregado, da forma de representação do conhecimento, do sistema de produções utilizado, do modelo de simulação, e , finalmente, das técnicas de planejamento, determinando, desta forma, uma arquitetura básica voltada para a solução do problema de planejamento associado ao controle e gerenciamento do tráfego de trens . Os parâmetros de classificação de cada método ou modelo seguiram a própria literatura , como, por exemplo, o trabalho de R. Slowinski [1986] que apresenta uma esquema de classificação de modelos de programação "fuzzy".

I.1 - Modelagem Matemática:

- modelo de "scheduling" e planejamento;
- tipo "job-shop" com restrições de capacidade;
- restrições estruturais : restrições de sequenciamento e restrições de precedência na forma disjuntiva;
- função objetiva : multi-objetiva associada ao tempo final de término das operações.

I.2 - Modelo de Programação "Fuzzy":

- problema multi-objetivo e multi-critério;
- formulação : Max-Min, inequações;
- função objetiva: "fuzzy" associada a funções de pertinência;
- funções de pertinência : linear por partes considerando: níveis de satisfação, níveis de prioridade e intervalos de suporte definidos segundo critérios;
- princípio de agregação: min, média algébrica;
- restrições flexíveis: função objetiva amarrada a funções de pertinência;
- modelo compensatório e interativo;
- solução: pontos eficientes ou não dominados.

I.3 - Método de Solução:

. processo de busca + raciocinio + satisfação de restrições;

. hipóteses básicas : sistema físico simbólico, espaço do problema, sistema de produção, métodos fracos, busca heurística , representação uniforme do conhecimento de longo termo, determinação de preferências em operadores;

. espaco do problema : arvore binária com profundidade igual ao número de conflitos;

. sistema físico simbólico: estado + operadores + função de avaliação;

. estado: tabela de tempos + função de avaliação + histórico dos conflitos + informações temporais;

. operadores: orientado segundo as restrições estruturais do problema e pela determinação de preferências. Voltado para a solução de conflitos na forma incremental (ordem cronológica);

. métodos fracos: A*, IDA*, busca em profundindade progressiva;

. heurística: consistente e admissível;

. função de avaliação: monotonicamente decrescente garantindo a solução otima;

. capacidade para processamento aproximado.

I.4 - Representação do Conhecimento:

. conhecimento factual, declarativo e relacionamento entre objetos : "frames";

. relacionamento ou propriedades de "frames": instanciação e heranca;

. conhecimento heurístico, intensivo, dependente do domínio: regras de produção;

. tipos de regras: análise, decisão, heurísticas, restrições, meta-regras, etc. ;

. conhecimento de curto termo (memória de trabalho): triplas objeto-atributo-valor;

. visualização e manipulação de "frames" ;

. atualização de informações dinâmicas .

I.5 - Sistema de Produção:

- organização e estrutura: hierárquica com modulos de regras;
- encadeamento: progressivo;
- método de inferência: ciclo reconhecimento-ação com "modus-ponens" e valores ativos ("frames");
- controle do conhecimento: solução de conflitos e meta-regras;
- memória de trabalho: dinâmica na forma O-A-V;
- regras com parâmetros;
- hipótese de sub-problemas ou espaços de problemas com sub-objetivos;
- interface : regras na forma textual, explanação de raciocínio;
- capacidade para raciocínio temporal.

I.6 - Modelo de Simulação:

- modelagem : entidades ou frames, atividades ou processos e eventos;
- programação e controle: exame de entidades com a realização de eventos garantidos e programação de eventos decisórios associados a ocorrência de conflitos;
- lista de eventos: tabela de tempos;
- tabela de tempos: instância particular dos eventos computados e eventos programados;
- inserção de eventos: não há;
- pesquisa de eventos decisórios: função polinomial da entrada (tabela de tempos);
- operações de acesso, cancelamento e programação de eventos: em tempo constante;
- progresso da simulação: seletivo, atraves da tomada de decisões;
- tomada de decisões: interação com sistema especialista;
- procedimento de otimização;
- possibilidade de retrocesso;
- interativo, possibilitando a visualização e manipulação de atributos de entidades;
- capacidade para visualização grafica;
- geração de variáveis "fuzzy" e aleatórias
- simulação paralela de eventos sem dependência lógica.

I.7 - Modelo de Planejamento:

- objetivo: conjunto de objetivos na forma conjuntiva considerando restrições temporais;
- estrutura de representação: tabela de tempos considerando o tempo de início mais cedo das atividades;
- processo de construção: planejamento hierárquico mais projeção e refinamentos;
- projeção: detecção de conflitos potenciais ou interação entre sub-problemas;
- passo de refinamento: adição de restrições disjuntivas para a solução de conflitos considerando o critério da verdade;
- algoritmo de controle: "deph first" ou "best first";
- produto: planos hierárquicos com ações parcialmente ordenados e ações paralelas conectadas sinergicamente;
- universo de planos: grafo de estados representando planos parciais;
- relação de preferencia: uso de uma função de avaliação heurística, com valores mapeados no intervalo (0.0..1.0);
- prova de dominância: admissibilidade e consistência da heurística;
- raciocínio temporal: baseado no cálculo de intervalos e na recuperação de informações;
- relacionamento entre intervalos: baseado em operações com base na representação de pontos ou instantes de tempo;
- manutenção de verdade: realização de inferências oportunistas , com base no uso de informações atualizadas e informações dependentes;
- organização: modular com abstrações funcionais e tipos abstratos de dados;
- execução: monitoração e replanejamento em tempo real

Apêndice II - Especificação e Organização do Programa SIGT

II.1 - Módulos Desenvolvidos para o Sistema SIGT:

```
DEFINITION MODULE TIME;

(* converte hora real para hora padrão e vice-versa *)

FROM CardinalIO IMPORT WriteCardinal, ReadCardinal;

IMPORT Terminal;

EXPORT    QUALIFIED    TipTime,    RealToTime,    TimeToReal,    WriteTime,
WriteRealTime;

TYPE TipTime = RECORD
                 hour: CARDINAL;
                 min: CARDINAL;
                 sec: CARDINAL;
                 cent: CARDINAL
               END;

PROCEDURE RealToTime(t: REAL; VAR CurTime: TipTime);
PROCEDURE TimeToReal(AuxTime: TipTime): REAL;
PROCEDURE WriteTime(AuxTime: TipTime);
PROCEDURE WriteRealTime(t: REAL);

END TIME.

DEFINITION MODULE STACK;

(* implementa uma pilha de inteiros como estrutura dinâmica *)

FROM SYSTEM IMPORT TSIZE;

FROM Storage IMPORT ALLOCATE, DEALLOCATE;

EXPORT  QUALIFIED  stack,  FacaVazia,  pop,  push,  vazia,  defina,  destrua,
retorna;

TYPE stack;

PROCEDURE FacaVazia(VAR s: stack);
PROCEDURE pop(VAR s: stack);
PROCEDURE vazia(s: stack): BOOLEAN;
PROCEDURE push(VAR s: stack; c: CARDINAL);
PROCEDURE defina(VAR s: stack);
PROCEDURE destrua(VAR s: stack);
PROCEDURE retorna(VAR s: stack; VAR c: CARDINAL);

END STACK.

DEFINITION MODULE LISTA;

(* implementa uma lista que armazena endereços, duplamente encadeada,
como  estrutura dinâmica *)
```

```
FROM SYSTEM IMPORT TSIZE, ADDRESS;

FROM Storage IMPORT ALLOCATE, DEALLOCATE;

FROM InOut IMPORT WriteLn, WriteString;

IMPORT Break;

EXPORT QUALIFIED list, empty, init, PutInit, PutEnd, select, inser,
take, destroy;

TYPE

  link = POINTER TO linkage;

  linkage = RECORD
            succ, pred : link;
            pinfo : ADDRESS;
            key: REAL
            END;

  list = RECORD
         head: link;
         tam, TamMax: CARDINAL;
         vazia, cheia: BOOLEAN
       END;

PROCEDURE init(VAR quee : list;tamanho: CARDINAL);
PROCEDURE  PutInit(VAR quee:  list;  aux:  ADDRESS;  VAR done:  BOOLEAN;
key:  REAL);
PROCEDURE PutEnd(VAR quee:  list;  aux:  ADDRESS;  VAR done :   BOOLEAN;
key:  REAL);
PROCEDURE   select(VAR  quee  :   list;   VAR  aux:   ADDRESS;VAR  done:
BOOLEAN;  VAR key:  REAL);
PROCEDURE inser( VAR quee:  list;  aux:  ADDRESS;  VAR  done:   BOOLEAN;
opcao:  BOOLEAN;  key:  REAL);
PROCEDURE empty( quee : list): BOOLEAN;
PROCEDURE take(VAR quee:  list;  VAR aux:  ADDRESS;  VAR done:  BOOLEAN;
key:  REAL);
PROCEDURE destroy ( VAR quee: list );

END LISTA.

DEFINITION MODULE STATIST;

(* implementa funcões estatísticas *)

FROM MathLib0 IMPORT sqrt, exp, real, entier;

FROM InOut IMPORT WriteLn;

EXPORT QUALIFIED
StatTip, RegTip, mean, variance, desviat, accum, InitStat, utilit,
InitStatreg, probab, regenerative, paramet, kolmo;

TYPE StatTip = RECORD
                 numer, sum, SumSqr : REAL;
```

```
                MinValue, MaxValue : REAL;
                prob : ARRAY  1..20  OF REAL
                END;

      RegTip = RECORD
                count, MaxQuee, MinQuee : CARDINAL;
                time, aux, cicle : REAL;
                numer, y, x : REAL;
                SumCicle, SumX, SumY : REAL;
                CicleVec, XVec, YVec : ARRAY  1..30  OF REAL
                END;

PROCEDURE mean(stat: StatTip): REAL;
PROCEDURE variance(stat: StatTip): REAL;
PROCEDURE desviat(stat: StatTip): REAL;
PROCEDURE accum(VAR stat: StatTip; value: REAL; weight: REAL);
PROCEDURE InitStat(VAR stat : StatTip);
PROCEDURE InitStatreg(VAR statreg: RegTip);
PROCEDURE utilit(stat: StatTip; capac: REAL): REAL;
PROCEDURE probab(stat: StatTip; value: REAL): REAL;
PROCEDURE  regenerative(VAR statreg: RegTip; clock: REAL; QueeSize:
CARDINAL);
PROCEDURE  paramet(statreg:  RegTip;  VAR ProbQuee, TimeMean, NumerMean,
EstVarian, ibound, iuper:  REAL;  cof:  REAL);
PROCEDURE kolmo(k, n:  CARDINAL;  VAR ax:  ARRAY OF CARDINAL;   kmaxpos,
kmaxneg, kmax:  REAL)

END STATIST.

DEFINITION MODULE GENERATO;

(* implementa funcões para a geracão de variáveis aleatórias *)

FROM MathLib0 IMPORT
ln, exp, sqrt, entier, real;

EXPORT  QUALIFIED  randum,  uniform,  expon,  normal, erlang, CriaEmpca,
empca,  poisson,  ParFuzzy,  geomet,  binomial,  weibull,  beta,  gamma,
lognorm;

PROCEDURE randum(VAR alea : REAL);
PROCEDURE ParFuzzy(par1,par2: REAL): REAL;
PROCEDURE uniform(par1, par2 : REAL): REAL;
PROCEDURE expon(par1 : REAL): REAL;
PROCEDURE normal(par1, par2 : REAL): REAL;
PROCEDURE erlang(par1, k : REAL): REAL;
PROCEDURE CriaEmpca(freq : ARRAY OF REAL; VAR freac : ARRAY OF REAL);
PROCEDURE empca(freac, val: ARRAY OF REAL): REAL;
PROCEDURE poisson(par1 : REAL): CARDINAL;
PROCEDURE lognorm(par1,par2 : REAL): REAL;
PROCEDURE gamma(par1, par2 : REAL): REAL;
PROCEDURE beta(par1, par2 : REAL): REAL;
PROCEDURE weibull(par1, par2 : REAL): REAL;
PROCEDURE binomial(par1, k : REAL): CARDINAL;
PROCEDURE geomet(par1 : REAL): CARDINAL;

END GENERATO.
```

```
DEFINITION MODULE SIMTREM;

(* simula parâmetros para a construção de instâncias de trens *)

FROM InOut IMPORT WriteLn, ReadCard, WriteString, Read;

FROM RealInOut IMPORT ReadReal, WriteReal;

FROM CardinalIO IMPORT ReadCardinal;

FROM GENERATO IMPORT randum, normal, ParFuzzy, expon, erlang, CriaEmpca,
empca, poisson, lognorm, gamma, beta;

FROM CLASSTREM IMPORT TipArray;

IMPORT Terminal;

EXPORT QUALIFIED SimulaTempos, SimulaClass, inicializa, NumTrem;

PROCEDURE SimulaTempos (VAR t: TipArray; si,sf: CARDINAL);
PROCEDURE SimulaClass (VAR c: CARDINAL);
PROCEDURE inicializa (ntc: CARDINAL);
PROCEDURE NumTrem (n: REAL): CARDINAL;

END SIMTREM.

DEFINITION MODULE CLASSTREM;

(* implementa "frame" classes de trens como um tipo abstrato com
estrutura dinâmica *)

IMPORT Break;

FROM SYSTEM IMPORT TSIZE;

FROM Storage IMPORT ALLOCATE, DEALLOCATE;

FROM DEFSECAO IMPORT si, sf;

EXPORT QUALIFIED PtClass, CriaClass, TipArray, TipCard, ModificaPrioDin;

TYPE PtClass = POINTER TO ObjClass;

     TipArray = ARRAY  1..24  OF REAL;

     TipCard = ARRAY  1..3  OF CARDINAL;

     ObjClass = RECORD
                  numero: CARDINAL;
                  nome: ARRAY  0..9  OF CHAR;
                  tempo: TipArray;
                  se,ss: CARDINAL;
                  PrioEst: CARDINAL;
                  PrioDin: CARDINAL;
                  comprimento: CARDINAL;
                  potencia: CARDINAL;
                  SecoesParada: TipCard
                END;
```

```
PROCEDURE CriaClass(cl:ARRAY OF CHAR; c:TipArray; a,b,d,e,f,g:CARDINAL;
                    sp:TipCard):PtClass;
PROCEDURE ModificaPrioDin (VAR Pc: PtClass; d: CARDINAL);

END CLASSTREM.

DEFINITION MODULE OBJTREM;

(* implementa "frame" trens como um tipo abstrato com estrutura dinâmica
*)

IMPORT Break;

FROM SYSTEM IMPORT TSIZE;

FROM Storage IMPORT ALLOCATE, DEALLOCATE;

FROM CLASSTREM IMPORT PtClass, TipArray;

FROM DEFSECAO IMPORT si, sf;

FROM DEFTREM IMPORT class;

FROM InOut IMPORT WriteString, Write, Read, WriteLn, ReadCard,
ReadString, WriteCard;

EXPORT QUALIFIED PtTrem, CriaTrem, RetornaTempos, sentido, DestroiTrem,
prioridade, RetornaTempo, RetornaSecFinal, codigo, comprimento,
potencia, parada, SuperClass, TempoEntrada, RetornaSecInicial, GravaT,
GravaC, NumClass, ModSlotTrem, PrSlotTrem, ModSlotClass, PrSlotClass,
PrioDinamica, RetornaSecInicialClass, RetornaSecFinalClass;


TYPE   TipCard  = ARRAY  1..3  OF CARDINAL;

     PtTrem;

PROCEDURE CriaTrem (pt:  PtClass;  t:  REAL;  s:  CHAR;  c,c1,c2:
CARDINAL ):  PtTrem;
PROCEDURE RetornaTempos (Pt: PtTrem; VAR t: TipArray);
PROCEDURE sentido(Pt: PtTrem): CARDINAL;
PROCEDURE SuperClass(Pt: PtTrem; VAR nome: ARRAY OF CHAR);
PROCEDURE prioridade(Pt: PtTrem): CARDINAL;
PROCEDURE PrioDinamica(Pt: PtTrem): CARDINAL;
PROCEDURE TempoEntrada(Pt: PtTrem): REAL;
PROCEDURE RetornaTempo(Pt: PtTrem; s: CARDINAL): REAL;
PROCEDURE RetornaSecFinal( Pt: PtTrem): CARDINAL;
PROCEDURE RetornaSecInicial(Pt:PtTrem): CARDINAL;
PROCEDURE comprimento(Pt: PtTrem): CARDINAL;
PROCEDURE RetornaSecFinalClass(Pt: PtTrem): CARDINAL;
PROCEDURE RetornaSecInicialClass(Pt:PtTrem): CARDINAL;
PROCEDURE codigo(Pt: PtTrem): CARDINAL;
PROCEDURE potencia(Pt: PtTrem): CARDINAL;
PROCEDURE NumClass(Pt: PtTrem): CARDINAL;
PROCEDURE DestroiTrem(Pt: PtTrem);
PROCEDURE parada(Pt: PtTrem; p:CARDINAL): CARDINAL;
PROCEDURE ModSlotTrem( VAR Pt: PtTrem; tex: ARRAY OF CHAR);
```

```
PROCEDURE PrSlotTrem( Pt: PtTrem; tex: ARRAY OF CHAR);
PROCEDURE ModSlotClass( VAR Pc: PtClass; tex: ARRAY OF CHAR);
PROCEDURE PrSlotClass( Pc: PtClass; tex: ARRAY OF CHAR);
PROCEDURE GravaT( Pt: PtTrem);
PROCEDURE GravaC( Pc: PtClass);

END OBJTREM.

DEFINITION MODULE DEFTREM;

(* implementa procedimentos para a criacao de instâncias de trens e
classes de trens, e manipulação de arquivos *)

IMPORT Break;

FROM MathLib0 IMPORT real;

FROM SIMTREM IMPORT SimulaTempos, SimulaClass, inicializa, NumTrem;

FROM OBJTREM IMPORT PtTrem, CriaTrem, DestroiTrem, sentido, GravaT,
GravaC;

FROM CLASSTREM IMPORT PtClass, CriaClass, TipArray, TipCard;

FROM InOut IMPORT WriteLn, ReadCard, WriteString, Read, ReadString,
Write, WriteCard;

FROM RealInOut IMPORT ReadReal, WriteReal;

FROM GENERATO IMPORT expon;

FROM DEFSECAO IMPORT si, sf, secao;

FROM OBJSECAO IMPORT HeadwayFinal, HeadwayInicial;

IMPORT Terminal;

EXPORT QUALIFIED trem, class, DefineTrens, nti, DefineClasses,
InsereTrem, DeleteTrem, ntt, ntc, SimulaTrens, GravaTrem, GravaClass;

VAR trem: ARRAY 1..16 OF PtTrem;
    nti, ntt, ntc: CARDINAL;
    class: ARRAY 1..10 OF PtClass;

PROCEDURE DefineTrens( nt: CARDINAL );
PROCEDURE SimulaTrens( nt: CARDINAL );
PROCEDURE DeleteTrem(n: CARDINAL);
PROCEDURE InsereTrem(pc,se,ss: CARDINAL; s: CHAR; t: REAL);
PROCEDURE DefineClasses(nc, cod: CARDINAL);
PROCEDURE GravaTrem();
PROCEDURE GravaClass();

END DEFTREM.

DEFINITION MODULE OBJSECAO;

(* implementa "frame" secoes como um tipo abstrato com estrutura
dinâmica *)
```

```
IMPORT Break;

FROM SYSTEM IMPORT TSIZE;

FROM Storage IMPORT ALLOCATE, DEALLOCATE;

FROM InOut IMPORT WriteString, WriteLn, Read, ReadString, Write,
ReadCard, WriteCard;

FROM RealInOut IMPORT ReadReal, WriteReal;

EXPORT QUALIFIED PtSecao, CriaSecaoSingela, CriaSecaoDupla,
ModificaStatus, NumLinhas, CapacPar, CapacImp, RestPar, RestImp,
StatusPar, StatusImp, numero, StatusPriPar, StatusPriImp, StatusSecPar,
StatusSecImp, DestroiSecao, ModSlotSecao, PrSlotSecao, NomeSecao,
HeadwayInicial, HeadwayFinal, TipStatus, GravaS, RestauraStatus,
comprimento;

TYPE PtSecao;

     TipStatus = (vazio, desativada, ocupada, desocupada);

PROCEDURE CriaSecaoSingela (s1,s2: TipStatus; n,n1,r1,r2,c1,c2,c3,c4,c5,
c6: CARDINAL; h1,h2: REAL; s: ARRAY OF CHAR): PtSecao;
PROCEDURE CriaSecaoDupla (s1,s2,s3,s4: TipStatus; n,n1,c3,c4: CARDINAL;
h1,h2: REAL;s: ARRAY OF CHAR): PtSecao;
PROCEDURE NumLinhas( Pt: PtSecao): CARDINAL;
PROCEDURE comprimento( Pt: PtSecao): CARDINAL;
PROCEDURE CapacPar( Pt: PtSecao): CARDINAL;
PROCEDURE CapacImp( Pt: PtSecao): CARDINAL;
PROCEDURE RestPar(Pt: PtSecao): CARDINAL;
PROCEDURE RestImp(Pt: PtSecao): CARDINAL;
PROCEDURE StatusPar (Pt: PtSecao): CARDINAL;
PROCEDURE StatusImp (Pt: PtSecao): CARDINAL;
PROCEDURE numero (Pt: PtSecao): CARDINAL;
PROCEDURE StatusPriPar (Pt: PtSecao): CARDINAL;
PROCEDURE StatusPriImp (Pt: PtSecao): CARDINAL;
PROCEDURE StatusSecPar (Pt: PtSecao): CARDINAL;
PROCEDURE StatusSecImp (Pt: PtSecao): CARDINAL;
PROCEDURE HeadwayFinal (Pt: PtSecao): REAL;
PROCEDURE HeadwayInicial (Pt: PtSecao): REAL;
PROCEDURE ModificaStatus (VAR Pt: PtSecao; tag: CARDINAL);
PROCEDURE RestauraStatus (VAR Pt: PtSecao; tag: CARDINAL);
PROCEDURE DestroiSecao (Pt: PtSecao);
PROCEDURE ModSlotSecao(VAR Pt: PtSecao; tex: ARRAY OF CHAR);
PROCEDURE PrSlotSecao(Pt: PtSecao; tex: ARRAY OF CHAR);
PROCEDURE NomeSecao(Pt: PtSecao; VAR ch: ARRAY OF CHAR);
PROCEDURE GravaS(Pt: PtSecao);

END OBJSECAO.

DEFINITION MODULE DEFSECAO;

(* implementa procedimentos para a criação de instâncias de seções e
manipulação de arquivos *)

FROM OBJSECAO IMPORT PtSecao, CriaSecaoSingela, CriaSecaoDupla,
```

```
DestroiSecao, TipStatus, GravaS;

FROM InOut IMPORT WriteLn, ReadCard, WriteString, Read, ReadString;

FROM RealInOut IMPORT ReadReal, WriteReal;

EXPORT  QUALIFIED secao, DefineSecoes, DeleteSecao, InsereSecao, si, sf,
GravaSecoes;

VAR secao: ARRAY  1..24  OF PtSecao;

     nts,si,sf: CARDINAL;

PROCEDURE DefineSecoes(ni,ns: CARDINAL);
PROCEDURE InsereSecao(n: CARDINAL);
PROCEDURE DeleteSecao(n: CARDINAL);
PROCEDURE GravaSecoes();

END DEFSECAO.

DEFINITION MODULE WorkMemory;

(* implementa memória de trabalho na forma O-A-V como um tipo  abstrato
com estrutura estática *)

IMPORT Break;

FROM SYSTEM IMPORT ADDRESS, TSIZE;

FROM Storage IMPORT ALLOCATE, DEALLOCATE;

FROM InOut IMPORT  WriteString, WriteLn, Read, Write;

IMPORT Terminal;

FROM CardinalIO IMPORT ReadCardinal, WriteCardinal;

EXPORT QUALIFIED objeto, cria, atributo, faca, remova, ModificaAtributo,
PrintAtributo,  LegValPref,  LegValParada,   LegValSent,   LegValStatus,
LegValConflito, LegValPrio, recency,*SetRec;

TYPE objeto;

      LegValConflito = (vazio1, ultrapassagem, cruzamento);

      LegValPref   =   (vazio2,  oper1,  oper2,  nenhum,  rejeicao,
                        RetardaP, RetardaS, inexiste);

      LegValParada = (vazio3, permitida, NaoPermitida, obrigatoria);

      LegValSent = (vazio4, par, impar);

      LegValStatus = (vazio5, desativada, ocupada, desocupada);

      LegValPrio = (vazio6,PoucImp,MedImp,import,MuitImp,critica,
                    inaceitavel);

PROCEDURE cria(VAR p: objeto; s: ARRAY OF CHAR; c: CARDINAL);
```

```
PROCEDURE atributo(a: CARDINAL; p: objeto): CARDINAL;
PROCEDURE faca (VAR p: objeto; a,c: CARDINAL);
PROCEDURE remova (VAR p: objeto);
PROCEDURE ModificaAtributo (VAR p: objeto; tex: ARRAY OF CHAR);
PROCEDURE PrintAtributo (p: objeto);
PROCEDURE recency(): CARDINAL;
PROCEDURE SetRec(x: CARDINAL);

END WorkMemory.

DEFINITION MODULE INTERPRETE;

(* implementa interpretador de módulos de regras  *)

FROM SYSTEM IMPORT ADDRESS;

FROM STACK IMPORT defina, pop, push, stack, FacaVazia, vazia;

FROM WorkMemory IMPORT recency, objeto;

FROM CardinalIO IMPORT WriteCardinal;

IMPORT Terminal;

EXPORT QUALIFIED When, And, Or, Then, IncLoop, InicializaLoop,
EsvaziaPilha, IniciaPilha, CondicaoTrue, PilhaRac, PilhaReg, TipOper,
TipProc, maior, menor , diferente, igual, SetNivel;

TYPE

TipOper = (GT,EQ,LT,NEQ);
TipProc = PROCEDURE (VAR objeto,CARDINAL,CARDINAL);

VAR

maior, menor, igual, diferente: TipOper;
PilhaRac, PilhaReg: stack;

PROCEDURE When (p: CARDINAL; oper: TipOper; c: CARDINAL; ValCrit,
n: CARDINAL);
PROCEDURE And (p: CARDINAL; oper: TipOper; c: CARDINAL;
 ValCrit: CARDINAL);
PROCEDURE Or (p: CARDINAL; oper: TipOper; c: CARDINAL;
 ValCrit: CARDINAL);
PROCEDURE Then (p: TipProc; pt: objeto; c,a,regra: CARDINAL);
PROCEDURE InicializaLoop(n1,n2,n3,n4,n5,n6,n8: CARDINAL;n9: BOOLEAN);
PROCEDURE IncLoop(VAR loop: CARDINAL; VAR bol: BOOLEAN);
PROCEDURE EsvaziaPilha();
PROCEDURE IniciaPilha();
PROCEDURE CondicaoTrue(): BOOLEAN;
PROCEDURE SetNivel();

END INTERPRETE.

DEFINITION MODULE MENU;

(* implementa funções básicas para a operação com janelas *)
```

```
FROM Graphics IMPORT Text, Line, GetWindow, Window, ClearWindow;

FROM Strings IMPORT Copy;

IMPORT Terminal;

IMPORT Break;

EXPORT QUALIFIED TipToggle, PutToggle, GetToggle, rect, ApagaWindow;


TYPE
               TipString = ARRAY  0..9  OF CHAR;

               TipToggle = RECORD
                   fix, mode: INTEGER;
                   cho, NumCho: CARDINAL;
                   strings: ARRAY  1..5  OF TipString;
                   loc: ARRAY  1..5  OF INTEGER
                 END;

PROCEDURE PutToggle(PrintToggle: TipToggle);
PROCEDURE GetToggle(VAR PrintToggle: TipToggle; prompt: ARRAY OF CHAR;
          y,x: INTEGER; VAR ChoResp: CARDINAL);
PROCEDURE rect(x1,y1,x2,y2,c: INTEGER);
PROCEDURE ApagaWindow(c,x1,y1,x2,y2: INTEGER);

END MENU.

DEFINITION MODULE EXPLAIN;

(* implementa funções básicas para a consulta de "frames",
elementos da memória de trabalho, leitura e visualização de regras
e explanação de raciocínio *)

FROM InOut IMPORT OpenInput, CloseInput, Done, ReadString, CloseOutput,
ReadCard, WriteLn, WriteCard, Read, Write, WriteString, EOL;

FROM Storage IMPORT ALLOCATE, DEALLOCATE;

FROM SYSTEM IMPORT TSIZE;

FROM LISTA IMPORT list, init, PutEnd, take, destroy;

FROM Strings IMPORT Assign, Pos, Length, CompareStr, Copy;

FROM STACK IMPORT retorna, pop;

FROM INTERPRETE IMPORT PilhaRac, PilhaReg;

FROM OBJTREM IMPORT ModSlotTrem, PrSlotTrem, ModSlotClass, PrSlotClass;

FROM DEFTREM IMPORT trem, class;

FROM OBJSECAO IMPORT ModSlotSecao, PrSlotSecao;

FROM DEFSECAO IMPORT secao;
```

```
FROM WorkMemory IMPORT ModificaAtributo, PrintAtributo, objeto;

FROM CardinalIO IMPORT ReadCardinal, WriteCardinal;

FROM MENU IMPORT TipToggle, PutToggle, GetToggle, ApagaWindow;

FROM Graphics IMPORT Line, ScreenMode;

IMPORT Terminal;

EXPORT QUALIFIED explain, consulta, ConsultaObjeto, ConstroiListaRegras;

PROCEDURE explain(c,p,t1,t2: CARDINAL);
PROCEDURE consulta;
PROCEDURE ConsultaObjeto(p: objeto);
PROCEDURE ConstroiListaRegras();

END EXPLAIN.

DEFINITION MODULE RuleSet0;

(* implementa módulo de regras no nível de análise *)

FROM INTERPRETE IMPORT When, And, Then, InicializaLoop, IncLoop,
IniciaPilha, maior, menor, igual, EsvaziaPilha;

FROM DEFTREM IMPORT trem;

FROM OBJTREM IMPORT sentido, prioridade;

FROM WorkMemory IMPORT objeto, faca, remova, cria, atributo,
ModificaAtributo, PrintAtributo, LegValConflito, LegValSent, LegValPref;

FROM DEFSECAO IMPORT secao;

FROM OBJSECAO IMPORT NumLinhas;

FROM RuleSet1 IMPORT PrefParar;

FROM RuleSet2 IMPORT PrefAvancar;

FROM RuleSet5 IMPORT PrefEncostar;

FROM RuleSet6 IMPORT PrefUltrapassar;

FROM RuleSet7 IMPORT PrefAguardar;

FROM EXPLAIN IMPORT explain;

EXPORT QUALIFIED preferencia;

PROCEDURE preferencia (PosSecao, t1, t2, cr: CARDINAL): CARDINAL;

END RuleSet0.

DEFINITION MODULE RuleSet1;

(* implementa módulo de regras que decide conflito de cruzamento em
```

```
linha singela  *)

FROM  INTERPRETE  IMPORT  When, And, Then, InicializaLoop,  IncLoop,
CondicaoTrue, maior, menor, igual, SetNivel, Or;

FROM DEFTREM IMPORT trem;

FROM OBJTREM IMPORT prioridade, PrioDinamica;

FROM  WorkMemory  IMPORT  objeto, ModificaAtributo, PrintAtributo, cria,
faca, atributo, remova, LegValPref, LegValParada, LegValPrio,recency,
SetRec;

FROM RuleSet3 IMPORT CondTremParar;

FROM TIMETAB IMPORT AguardaTrem;

EXPORT QUALIFIED PrefParar;

PROCEDURE PrefParar (PosSecao, t1, t2: CARDINAL): CARDINAL;

END RuleSet1.

DEFINITION MODULE RuleSet2;

(*  implementa  módulo  de  regras  que define conflito de cruzamento em
linha dupla *)

IMPORT Break;

FROM  INTERPRETE  IMPORT  When, And, Then, InicializaLoop,  IncLoop,
CondicaoTrue, maior, menor, igual, SetNivel;

FROM DEFTREM IMPORT trem;

FROM OBJTREM IMPORT prioridade;

FROM  WorkMemory  IMPORT  objeto,  faca,  remova,  cria,  atributo,
ModificaAtributo, PrintAtributo, LegValPref, LegValParada,  recency,
SetRec;

FROM RuleSet4 IMPORT CondAvancarTremP, CondAvancarTremS;

EXPORT QUALIFIED PrefAvancar;

PROCEDURE PrefAvancar (PosSecao, t1, t2: CARDINAL): CARDINAL;

END RuleSet2.

DEFINITION MODULE RuleSet3;

(* implementa módulo de regras que decide condição de trem parar *)

FROM  INTERPRETE  IMPORT  When, And, Then, InicializaLoop, IncLoop,
CondicaoTrue, maior, menor, igual, SetNivel;

FROM DEFTREM IMPORT trem;
```

```
FROM TIMETAB IMPORT AguardaTrem;

FROM OBJTREM IMPORT sentido, comprimento, parada, potencia, prioridade;

FROM WorkMemory IMPORT objeto, ModificaAtributo, PrintAtributo, cria,
faca, atributo, remova, LegValParada, LegValSent, LegValStatus, recency,
SetRec;

FROM DEFSECAO IMPORT secao;

FROM OBJSECAO IMPORT NumLinhas, CapacPar, CapacImp, StatusPar,
StatusImp, numero, RestPar, RestImp;

EXPORT QUALIFIED CondTremParar;

PROCEDURE CondTremParar (PosSecao, x: CARDINAL): CARDINAL;

END RuleSet3.

DEFINITION MODULE RuleSet4;

(* implementa módulo de regras que decide a condição do trem avancar  *)

IMPORT Break;

FROM  INTERPRETE IMPORT When, And, Then, InicializaLoop, IncLoop,
CondicaoTrue ,maior, igual, menor, SetNivel;

FROM DEFTREM IMPORT trem;

FROM OBJTREM IMPORT sentido;

FROM WorkMemory IMPORT objeto, faca, remova, cria, atributo,
ModificaAtributo, PrintAtributo, LegValParada, LegValSent, LegValStatus,
recency, SetRec;

FROM DEFSECAO IMPORT secao;

FROM  OBJSECAO  IMPORT  StatusPriPar,  StatusPriImp,  StatusSecPar,
StatusSecImp;

EXPORT QUALIFIED CondAvancarTremP, CondAvancarTremS;

PROCEDURE CondAvancarTremP (PosSecao, x: CARDINAL): CARDINAL;

PROCEDURE CondAvancarTremS (PosSecao, x: CARDINAL): CARDINAL;

END RuleSet4.

DEFINITION MODULE RuleSet5;

(* implementa módulo de regras que decide ultrapassagem em linha singela
*)

FROM INTERPRETE IMPORT When, And, Or, Then, InicializaLoop, IncLoop,
IniciaPilha, maior, menor, igual, CondicaoTrue, SetNivel;

FROM DEFTREM IMPORT trem;
```

```
FROM OBJTREM IMPORT prioridade, sentido;

FROM DEFSECAO IMPORT secao;

FROM OBJSECAO IMPORT StatusPar, StatusImp, NumLinhas;

FROM WorkMemory IMPORT objeto, faca, remova, cria, atributo,
ModificaAtributo, PrintAtributo, LegValParada, LegValPref, LegValSent,
recency, SetRec, LegValStatus, LegValPrio;

FROM RuleSet3 IMPORT CondTremParar;

EXPORT QUALIFIED PrefEncostar;

PROCEDURE PrefEncostar (PosSecao, t1, t2: CARDINAL): CARDINAL;

END RuleSet5.

DEFINITION MODULE RuleSet6;

(* implementa módulo de regras que decide ultrapassagem em linha dupla
*)

IMPORT Break;

FROM INTERPRETE IMPORT When, And, Then, InicializaLoop, IncLoop,
CondicaoTrue, maior, menor, igual, SetNivel;

FROM DEFTREM IMPORT trem;

FROM OBJTREM IMPORT prioridade, sentido;

FROM DEFSECAO IMPORT secao;

FROM OBJSECAO IMPORT StatusPar, StatusImp, NumLinhas;

FROM WorkMemory IMPORT objeto, faca, remova, cria, atributo,
ModificaAtributo, PrintAtributo, LegValPref, LegValParada, LegValStatus,
recency, SetRec, LegValSent;

FROM RuleSet4 IMPORT CondAvancarTremP, CondAvancarTremS;

EXPORT QUALIFIED PrefUltrapassar;

PROCEDURE PrefUltrapassar (PosSecao, t1, t2: CARDINAL): CARDINAL;

END RuleSet6.

DEFINITION MODULE RuleSet7;

(* implementa módulo de regras que decide conflito de manutenção do
"headway" *)

FROM INTERPRETE IMPORT When, And, Or, Then, InicializaLoop, IncLoop,
IniciaPilha, maior, menor, igual, diferente, CondicaoTrue, SetNivel;

FROM DEFTREM IMPORT trem;
```

```
FROM OBJTREM IMPORT prioridade, sentido;

FROM DEFSECAO IMPORT secao;

FROM OBJSECAO IMPORT StatusPar, StatusImp, NumLinhas;

FROM WorkMemory IMPORT objeto, faca, remova, cria, atributo,
ModificaAtributo, PrintAtributo, LegValParada, LegValPref, LegValSent,
recency, SetRec, LegValStatus, LegValPrio;

FROM RuleSet3 IMPORT CondTremParar;

EXPORT QUALIFIED PrefAguardar;

PROCEDURE PrefAguardar (PosSecao, t1, t2: CARDINAL): CARDINAL;

END RuleSet7.


DEFINITION MODULE CALC;

(* implementa cálculo da função de avaliação, construção e visualização
gráfica das funções de pertinência , definição e redefinição de
objetivos *)

FROM RealInOut IMPORT ReadReal, WriteReal;

FROM InOut IMPORT OpenInput, CloseInput, ReadCard, WriteLn, WriteString,
WriteCard, Done;

FROM MathLib0 IMPORT entier;

FROM TIMETAB IMPORT TipTimeTable, DeterminaAtraso;

FROM DEFTREM IMPORT trem, class, ntt, ntc;

FROM OBJTREM IMPORT RetornaSecFinal, RetornaSecInicial, NumClass,
RetornaTempos;

FROM CLASSTREM IMPORT TipArray, ModificaPrioDin;

FROM Graphics IMPORT Line, ScreenMode, Text;

FROM Strings IMPORT Copy;

FROM RealConversions IMPORT RealToString;

FROM NumberConversion IMPORT CardToString;

FROM MENU IMPORT GetToggle, PutToggle, TipToggle, rect;

IMPORT Terminal;

EXPORT QUALIFIED CalcFa,CalcLimites,IniciaFuncPert,vet, AvaliaFunc,
RedefineObj, DisplaiFunc, AtrasoFunc;

VAR vet: ARRAY 1..6 OF REAL;
```

```
PROCEDURE CalcFa (VAR u: TipArray; t1: CARDINAL; t: REAL;
 op: CARDINAL): REAL;
PROCEDURE CalcLimites (tab: TipTimeTable);
PROCEDURE IniciaFuncPert( VAR u: TipArray);
PROCEDURE AvaliaFunc(t: REAL; VAR f: TipArray; AuxTab: TipTimeTable);
PROCEDURE RedefineObj(u: TipArray);
PROCEDURE DisplaiFunc();
PROCEDURE AtrasoFunc(t1: CARDINAL; fp: REAL): REAL;

END CALC.

DEFINITION MODULE GRADE;

(* implementa função para a visualização gráfica da grade de trens *)

IMPORT Break;

FROM Strings IMPORT Copy;

FROM RealConversions IMPORT RealToString;

FROM NumberConversion IMPORT CardToString;

FROM Graphics IMPORT ScreenMode, Line, Text, Window, ClearWindow,
GetWindow;

FROM CLASSTREM IMPORT TipArray;

FROM DEFTREM IMPORT trem, ntt, nti;

FROM OBJTREM IMPORT sentido, RetornaTempo, RetornaSecFinal,
RetornaSecInicial;

FROM DEFSECAO IMPORT secao;

FROM OBJSECAO IMPORT comprimento, NomeSecao, NumLinhas;

FROM TIMETAB IMPORT TipTimeTable, TabTempos;

FROM STATE IMPORT RetiraTab, RetiraFuncPert, transfere, cria, PtState;

FROM ALGORITM IMPORT EstadoFinal;

FROM MathLib0 IMPORT real, entier;

FROM RealInOut IMPORT ReadReal;

FROM InOut IMPORT ReadCard, WriteLn, WriteString;

FROM MENU IMPORT TipToggle, PutToggle, GetToggle, rect, ApagaWindow;

FROM CALC IMPORT AtrasoFunc;

IMPORT Terminal;

EXPORT QUALIFIED grade;
```

```
PROCEDURE grade(c,m: INTEGER; s1,s2: CARDINAL; r1,r2: REAL;
 EstadoAux: PtState);

END GRADE.


DEFINITION MODULE TIMETAB;

(* implementa tabela de tempos como um tipo abstrato com estrutura
estática *)

IMPORT Break;

FROM OBJTREM IMPORT RetornaTempo, prioridade, RetornaTempos,
TempoEntrada, sentido, RetornaSecInicial, RetornaSecFinal;

FROM OBJSECAO IMPORT NumLinhas, HeadwayFinal, HeadwayInicial;

FROM DEFTREM IMPORT trem, ntt, nti;

FROM DEFSECAO IMPORT secao, si, sf;

FROM InOut IMPORT ReadString, ReadCard, WriteLn, WriteCard, WriteString;

FROM CLASSTREM IMPORT TipArray;

FROM GENERATO IMPORT ParFuzzy;

EXPORT QUALIFIED TipTimeTable, DeterminaProxConflito, AtualizaTimeTable,
InicializaTimeTable, DeterminaAtraso, VerificaConflito,
ReconstroiTimeTable, AguardaTrem, SetTremAguarda, TabTempos;

TYPE TipTimeTable = ARRAY 1..16 OF ARRAY 0..25 OF REAL;

VAR TabTempos: TipTimeTable;

PROCEDURE DeterminaProxConflito (tab: TipTimeTable; VAR PosSecao,trem1,
          trem2: CARDINAL; VAR instante: REAL; VAR conflito: BOOLEAN);
PROCEDURE VerificaConflito (tab: TipTimeTable): BOOLEAN;
PROCEDURE DeterminaAtraso (tab: TipTimeTable; t1: CARDINAL): REAL;
PROCEDURE AtualizaTimeTable (VAR tab: TipTimeTable; VAR t1,t2: CARDINAL;
                             ps,oper: CARDINAL);
PROCEDURE InicializaTimeTable (VAR tab: TipTimeTable; r: REAL);
PROCEDURE ReconstroiTimeTable (VAR tab: TipTimeTable; tempo: REAL);
PROCEDURE AguardaTrem(t,p: CARDINAL): CARDINAL;
PROCEDURE SetTremAguarda(t,p: CARDINAL; b: BOOLEAN);

END TIMETAB.

DEFINITION MODULE STATE;

(* implementa estado como um tipo abstrato com estrutura dinâmica,
operadores e procedimentos para manutenção de verdade e cálculo de
intervalos *)

IMPORT Break;

FROM SYSTEM IMPORT TSIZE;
```

```
FROM Storage IMPORT ALLOCATE, DEALLOCATE;

FROM TIMETAB IMPORT TipTimeTable, InicializaTimeTable,
AtualizaTimeTable, DeterminaProxConflito, SetTremAguarda, AguardaTrem;

FROM CALC IMPORT IniciaFuncPert, CalcLimites, CalcFa, vet, DisplaiFunc;

FROM RuleSet0 IMPORT preferencia;

FROM CLASSTREM IMPORT TipArray;

FROM OBJSECAO IMPORT TipStatus, ModificaStatus, NumLinhas, CapacPar,
CapacImp, RestauraStatus, HeadwayInicial;

FROM OBJTREM IMPORT prioridade, NumClass, sentido, RetornaTempo,
RetornaSecFinal;

FROM DEFSECAO IMPORT secao, si, sf;

FROM DEFTREM IMPORT trem, ntt, nti;

FROM InOut IMPORT WriteLn, WriteString, WriteCard;

FROM RealInOut IMPORT ReadReal, WriteReal;

FROM MemExt IMPORT DelTab, GravaTab,RecuperaTab,CriaExtMem,FechaExtMem;

IMPORT Terminal;

EXPORT QUALIFIED PtState, GeraRaiz, ImpUlt, match, GeraEstado, fim,
InicializaRaiz, operador, RetiraTab, RetiraConf, RetiraFuncPert,
EstadoValFun, transfere, cria, destrua, ImpAtraso, print, poda,
SetAguarda;

TYPE PtState = POINTER TO TipState;

     TipIntervalo = RECORD  cheg, said: REAL  END;

     TipState = RECORD
                  high, low: CARDINAL;
                  ptr: ADDRESS;
                  nivel: CARDINAL;
                  ValFun: REAL;
                  FuncPert: TipArray;
                  conflito: ARRAY [0..60] OF RECORD
                              trem1,trem2,NumSecao,CodAcao: CARDINAL;
                              instante: REAL
                            END;
                  aguarda: ARRAY [0..60] OF RECORD
                                              trem,secao: CARDINAL
                                            END;
                  bloqueio: ARRAY [1..2] OF RECORD
                                              trem,cod: CARDINAL;
                                              janela: TipIntervalo;
                                              PosSecao: CARDINAL
                                            END
                END;
```

```
PROCEDURE GeraRaiz( VAR raiz: PtState; c: CARDINAL; r: REAL);
PROCEDURE InicializaRaiz(VAR raiz: PtState; TabRaiz: TipTimeTable);
PROCEDURE operador(VAR estado: PtState; VAR t1,t2,ps: CARDINAL;
VAR t: REAL; VAR oper: CARDINAL; sr,cr: CARDINAL);
PROCEDURE GeraEstado (estado: PtState; VAR EstadoGerado: PtState;
                      t1,t2,ps: CARDINAL; t: REAL; oper,op: CARDINAL);
PROCEDURE fim (estado: PtState; h: REAL):BOOLEAN;
PROCEDURE RetiraTab(EstadoAux: PtState; VAR TabAux: TipTimeTable);
PROCEDURE RetiraConf(EstadoAux: PtState; VAR t1,t2,ps,op,n: CARDINAL;
 VAR inst: REAL);
PROCEDURE RetiraFuncPert(EstadoAux: PtState; VAR u: TipArray);
PROCEDURE EstadoValFun(EstadoAux: PtState): REAL;
PROCEDURE match(EstadoAux: PtState; t1,t2,ps,n: CARDINAL;
 VAR acao,pos: CARDINAL; oper: CARDINAL; VAR rep: BOOLEAN);
PROCEDURE transfere (VAR ori: PtState; dest: PtState);
PROCEDURE cria(VAR EstadoAux: PtState);
PROCEDURE destrua(VAR EstadoAux: PtState);
PROCEDURE ImpAtraso(EstadoIni,EstadoFim: PtState);
PROCEDURE poda(EstadoGerado: PtState; n: CARDINAL): BOOLEAN;
PROCEDURE ImpUlt(PrintEstado: PtState; c,c1,c2: CARDINAL);
PROCEDURE print(n,c1,c2: CARDINAL);
PROCEDURE SetAguarda(VAR EstadoAux: PtState; n: CARDINAL);

END STATE.

DEFINITION MODULE ALGORITM;

(* implementa procedimentos para a realização de buscas heurísticas *)

IMPORT Break;

FROM InOut IMPORT WriteLn, WriteString, OpenOutput, CloseOutput, Done;

FROM LISTA IMPORT list, PutInit, PutEnd, select, inser, init, destroy;

FROM  STATE IMPORT ImpAtraso, RetiraTab, print, operador, PtState, cria,
ImpUlt, destrua, GeraEstado, transfere,  poda,  EstadoValFun,  GeraRaiz,
fim;

FROM TIMETAB IMPORT VerificaConflito, TipTimeTable;

FROM OBJTREM IMPORT RetornaTempo, sentido;

FROM DEFTREM IMPORT trem, ntt, nti;

FROM CALC IMPORT vet;

EXPORT    QUALIFIED    EstadoFinal,    AEstrela,    ProgressiveDeepening,
RecuperaMemoria, InicializaEstado, IDAEstrela;

VAR EstadoFinal: PtState;

PROCEDURE AEstrela (HoraLimite: REAL; n3,SeletorBusca,MetMat,op,sr,cr:
          CARDINAL; lim: BOOLEAN; LimAceitavel: REAL);
PROCEDURE ProgressiveDeepening (HoraLimite: REAL; n3, SeletorBusca,
                     MetMat, op, sr, cr: CARDINAL; LimAceitavel: REAL);
PROCEDURE RecuperaMemoria();
PROCEDURE InicializaEstado(EstadoAux: PtState);
```

```
PROCEDURE IDAEstrela (HoraLimite: REAL; n3,op,sr,cr: CARDINAL);

END ALGORITM.

DEFINITION MODULE SIMULA;

(* implementa a simulação do planejamento, processo de monitoração,
processo de replanejamento e o módulo principal do modelo de simulação
*)

FROM PROCESSES IMPORT SIGNAL, Init, SEND, WAIT, Awaited, Terminate;

FROM LISTA IMPORT list, init, PutInit, select, destroy;

FROM RealInOut IMPORT ReadReal, WriteReal;

FROM GRADE IMPORT grade;

FROM Graphics IMPORT Text, ScreenMode, Palette, Line, Window, GetWindow,
ClearWindow;

FROM RealConversions IMPORT RealToString;

FROM NumberConversion IMPORT IntToString, CardToString;

FROM CardinalIO IMPORT ReadCardinal, WriteCardinal;

FROM InOut IMPORT OpenOutput, CloseOutput, WriteCard, Done;

FROM TIMETAB IMPORT ReconstroiTimeTable, TipTimeTable;

FROM CALC IMPORT AvaliaFunc, RedefineObj, AtrasoFunc;

FROM TIME IMPORT WriteRealTime;

FROM TimeDate IMPORT GetTime, Time;

FROM STATE IMPORT PtState, GeraEstado, fim, operador,RetiraTab,
ImpAtraso, RetiraConf, GeraRaiz, ImpUlt, match, RetiraFuncPert,
transfere, cria, destrua, InicializaRaiz, SetAguarda;

FROM ALGORITM IMPORT AEstrela, ProgressiveDeepening, RecuperaMemoria,
EstadoFinal, InicializaEstado;

FROM CLASSTREM IMPORT TipArray;

FROM OBJTREM IMPORT codigo, SuperClass, sentido, prioridade;

FROM OBJSECAO IMPORT NomeSecao, NumLinhas;

FROM EXPLAIN IMPORT consulta;

FROM DEFTREM IMPORT trem, ntt, nti;

FROM DEFSECAO IMPORT secao, si, sf;

FROM Strings IMPORT Copy, Assign;
```

```
FROM MENU IMPORT TipToggle, PutToggle, GetToggle, ApagaWindow, rect;

IMPORT Break;

IMPORT Terminal;

EXPORT QUALIFIED RePlan, rep, SimuPlan, MoniPlan, SimuAcao, SetHora;

VAR rep: SIGNAL;

PROCEDURE RePlan(VAR termino: BOOLEAN);
PROCEDURE SimuPlan();
PROCEDURE MoniPlan();
PROCEDURE SimuAcao();
PROCEDURE SetHora(h0,h1,h2,h3: REAL);

END SIMULA.

DEFINITION MODULE PROCESSES;

( * implementa primitivas de sincronização de processos baseado
na utilização de semáforos com contadores *)

FROM SYSTEM IMPORT
ADDRESS, PROCESS, TSIZE, NEWPROCESS, TRANSFER;

FROM Storage IMPORT  ALLOCATE;

FROM LISTA IMPORT init, PutEnd, select, empty, list;

IMPORT Terminal;

EXPORT QUALIFIED  nullprocess, SIGNAL, processid, idle, currentprocess,
Init, SEND, WAIT, Awaited, StartProcess, Terminate, equal;

TYPE   SIGNAL; processid;

VAR    nullprocess: processid;
       idle: SIGNAL;
       currentprocess: processid;

PROCEDURE Init (VAR s:SIGNAL (* out *));
PROCEDURE SEND (VAR s:SIGNAL (* in/out *));
PROCEDURE WAIT (VAR s:SIGNAL (* in/out *));
PROCEDURE Awaited (s:SIGNAL (* in *)):BOOLEAN;
PROCEDURE StartProcess (p:PROC (* in *); wssize:CARDINAL (* in *));
PROCEDURE Terminate;
PROCEDURE equal (p1:processid (* in *); p2:processid (* in *)):BOOLEAN;

END PROCESSES.

DEFINITION MODULE MemExt;

(* implementa acesso a memória extendida através da organização de um
 arquivo de registros em um drive virtual *)
FROM Storage IMPORT ALLOCATE, DEALLOCATE;

FROM FileSystem IMPORT File, Create, Close, Lookup, Reset, Length,
```

```
SetRead, SetWrite, SetPos, GetPos, ReadNBytes, WriteNBytes, Delete;

FROM RealConversions IMPORT RealToString, StringToReal;

FROM SYSTEM IMPORT TSIZE, ADDRESS;

FROM TIMETAB IMPORT TipTimeTable;

EXPORT QUALIFIED CriaExtMem, FechaExtMem, DelTab, RecuperaTab, GravaTab;

PPROCEDURE CriaExtMem();
PROCEDURE FechaExtMem();
PROCEDURE DelTab(lo,hi: CARDINAL;ender:ADDRESS);
PROCEDURE RecuperaTab(VAR tab: TipTimeTable; nt,nsi,nsf,lo,hi: CARDINAL;
                      ender: ADDRESS);
PROCEDURE GravaTab(VAR ender: ADDRESS; VAR lo,hi:CARDINAL; nt,nsi,nsf:
                   CARDINAL; tab: TipTimeTable;cr: CARDINAL);

END MemExt.


MODULE SIGT;

(* módulo principal  do  Sistema  Inteligente  para  o  Gerenciamento de
Tráfego Ferroviário  *)

IMPORT DebugPMD;

IMPORT Break;

FROM Graphics IMPORT ScreenMode, Text, Palette, Window, ClearWindow;

FROM ALGORITM IMPORT AEstrela, IDAEstrela, ProgressiveDeepening,
 RecuperaMemoria;

FROM EXPLAIN IMPORT  consulta, ConstroiListaRegras;

FROM DEFSECAO IMPORT DefineSecoes, InsereSecao, DeleteSecao,GravaSecoes;

FROM DEFTREM IMPORT DefineTrens, SimulaTrens, InsereTrem, DeleteTrem,
DefineClasses, GravaTrem, GravaClass;

FROM PROCESSES IMPORT Init, idle, WAIT, SEND, StartProcess;

FROM SIMULA IMPORT MoniPlan, RePlan, SimuPlan, SimuAcao, SetHora, rep;

FROM Strings IMPORT CompareStr;

IMPORT Terminal;

FROM InOut IMPORT OpenInput, Done, CloseInput, OpenOutput, CloseOutput,
ReadCard, Read, ReadString, ReadInt, WriteLn;

FROM RealInOut IMPORT ReadReal;

FROM GRADE IMPORT grade;

VAR ch: CHAR;
```

```
    cod,nc, si, sf, nt: CARDINAL;
    arq: ARRAY [1..3] OF CHAR;
    h,h2: REAL;
    Sim,Plan: BOOLEAN;
    mode, color, pal: INTEGER;

PROCEDURE displai(tex: ARRAY OF CHAR);
PROCEDURE AbraArquivoLeitura(tex: ARRAY OF CHAR; ext: ARRAY OF CHAR);
PROCEDURE FechaArquivoLeitura();
PROCEDURE AbraArquivoGravacao(tex: ARRAY OF CHAR; ext: ARRAY OF CHAR);
PROCEDURE FechaArquivoGravacao();
PROCEDURE secoes();
PROCEDURE class();
PROCEDURE trem();
PROCEDURE LeRegras();
PROCEDURE IniciaBusca();
PROCEDURE insere();
PROCEDURE delete();
PROCEDURE ConsultaFrames();
PROCEDURE ImpGrade();
PROCEDURE planejamento();
PROCEDURE simulacao();
PROCEDURE GravaArquivo();
PROCEDURE AguardaReplan();
PROCEDURE GetMode();

END SIGT.
```

II.2 - Módulos da Biblioteca Modula-2:

Módulos que implementam funções de I/O e Arquivos:
Terminal, CardinalIO, RealInOut, InOut, FileSystem

Módulo para manipulação de "strings":
Strings

Módulo para o gerenciamento de memória de "heap"
Storage

Módulo para a manipulação do relógio:
TimeDate

Módulo que implementa funções matemáticas:
MathlibO

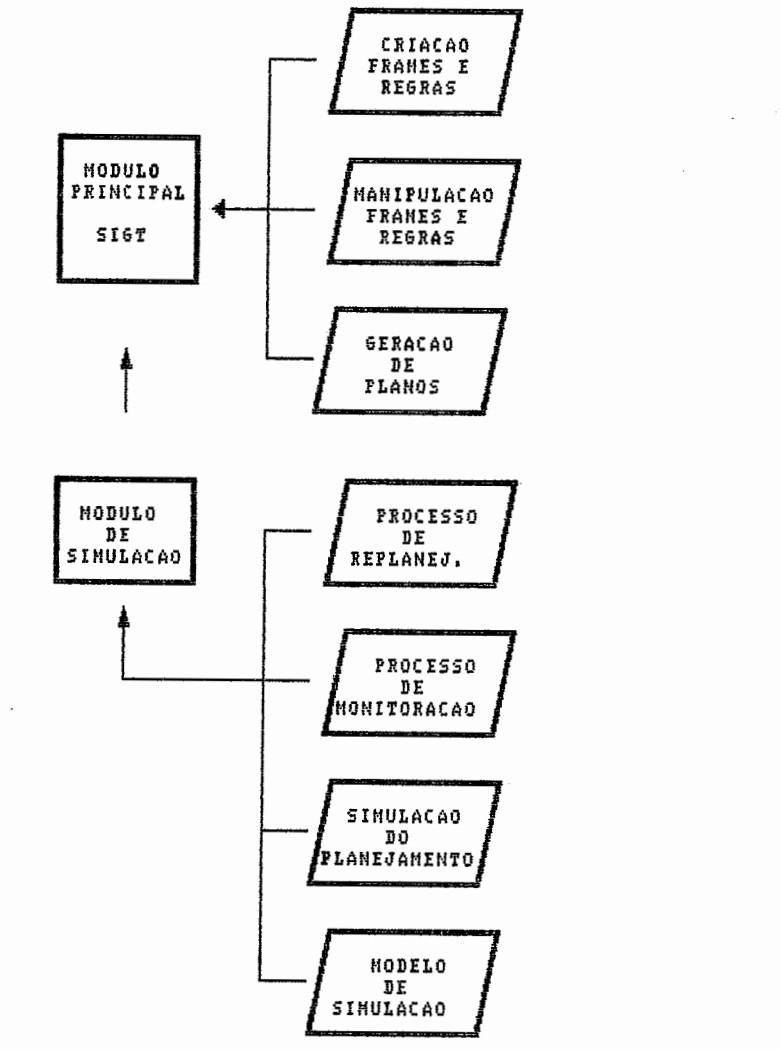
Módulo que implementa rotinas gráficas:
Graphics

Módulo para o cancelamento do programa:
Break, DebugPMD

Módulos para conversão de tipos:
RealConversion, NumberConversion

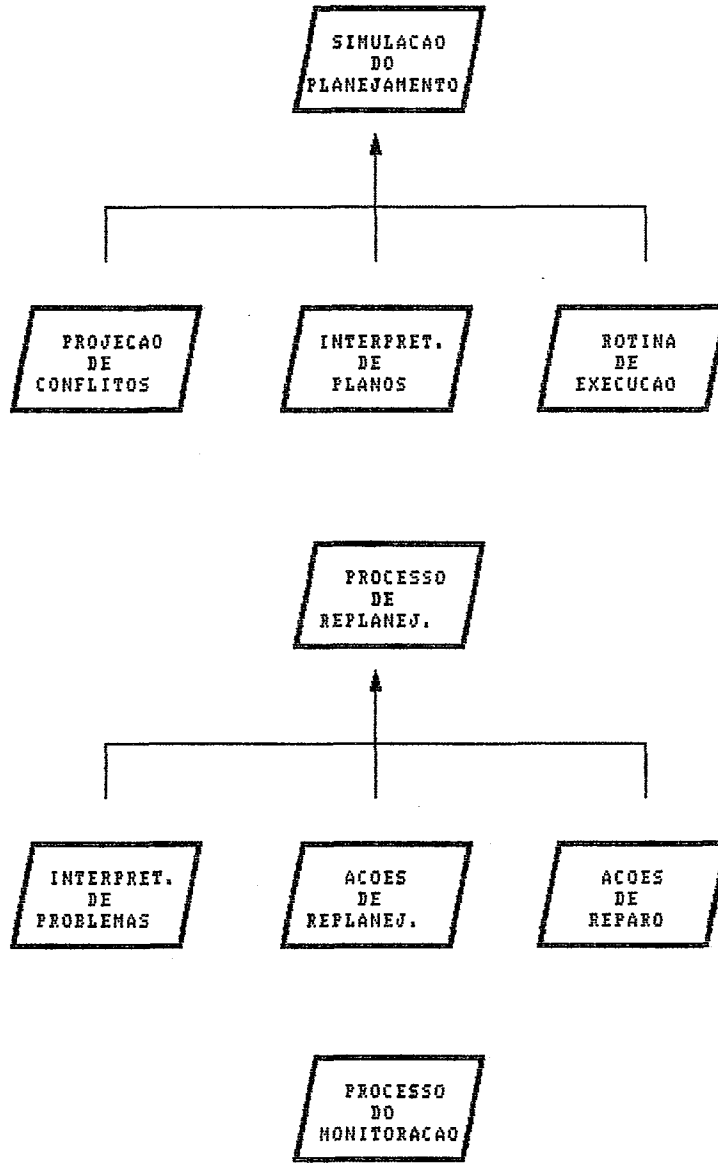
Módulo do sistema Modula-2 (funções e tipos de baixo nível):
SYSTEM

II.3 - Diagrama de Fluxos do Programa SIGT

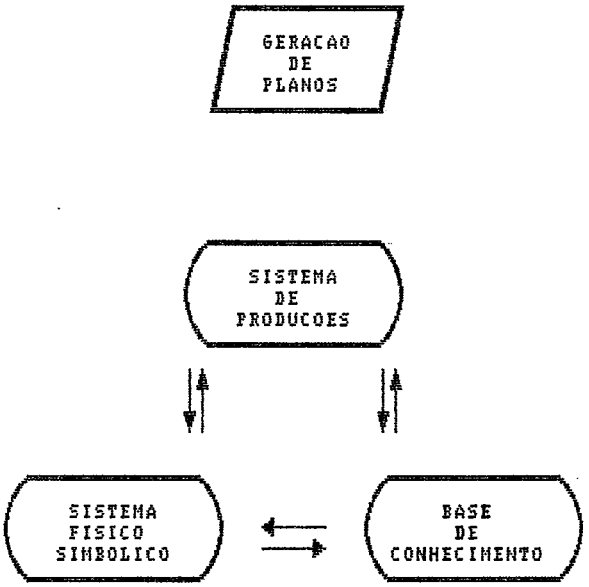


Convencao :

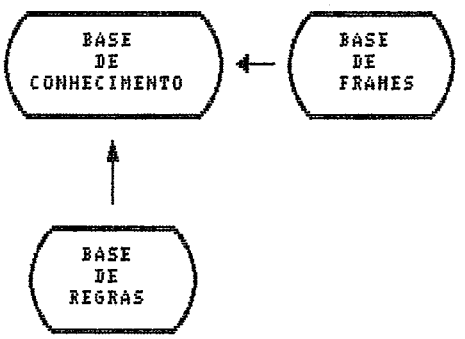
Execucao de Planos



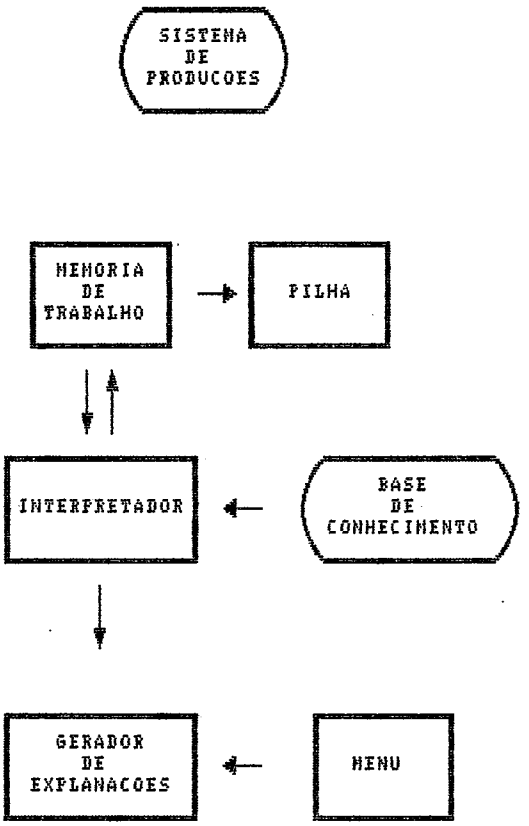
Geracao de Planos



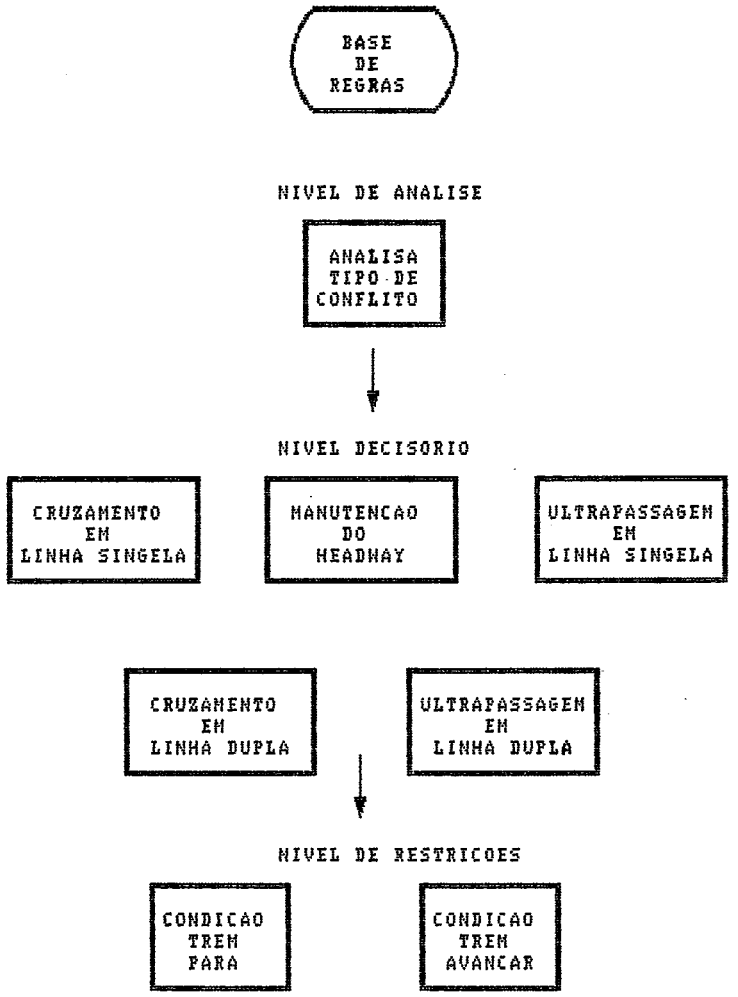
Base de Conhecimento



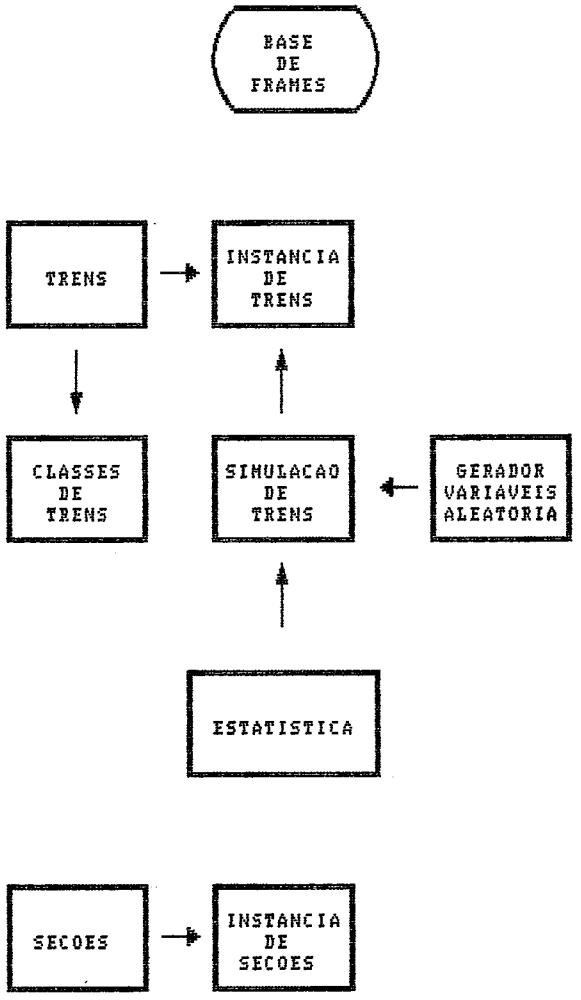
Sistema Baseado em Conhecimento



Modulos de Regras



Base de Frames



Sistema Fisico Simbolico