PLANEJAMENTO DO "START-UP" DE PLANTAS QUÍMICAS ATRAVÉS DE REDES DE PETRI E ALGORITMOS GENÉTICOS

Márcia Aparecida Fernandes

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM 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. Luís Alfredo Vidal de Carvalho, D.Sc.
(Presidente)

Prof. Felix Mora-Camino, Ph.D.

Prof. Carlos Augusto G. Perlingeiro, D. Sc.

Prof. João Nunes de Souza, D. Sc.

Prof. Jules Slama, Ph. D.

Prof. Luiz Otávio de Azevedo, D. Sc.

RIO DE JANEIRO, RJ - Brasil
Outubro de 1996

Fernandes, Márcia Aparecida

(Rio de Janeiro) 1996.

VIII( ), 131 p., 29.7 cm (COPPE/UFRJ, D. Sc., Engenharia de Sistemas e Computação, 1996)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Síntese de Procedimentos Operacionais 2. Algoritmos Genéticos 3. Planejamento 4. Sistemas Híbridos 5. Redes de Petri

I. COPPE/UFRJ II.Título (Série)

A todos que me ajudaram

Agradecimentos

Aos professores Luís Alfredo Vidal de Carvalho (orientador), Felix Mora-Camino (orientador), Valmir Carneiro Barbosa e Mário Benevides, orientadores do exame de qualificação. E também aos professores Valette e Aguilar.

À minha família, a Clícia e família, a Edna e família, a Luciene e a Jean-Marc.

Por fim, agradeço ao CNPq, a todos os colegas e funcionários da Universidade Federal de Uberlândia e da Universidade Federal do Rio de Janeiro, por auxílios como bolsa de estudo, afastamento das minhas funções de professora no Departamento de Informática e enfim, pelo apoio moral e técnico durante todo o período deste trabalho.

Agradeço ainda, ao Laboratoire d'Analise et Archicteture de Sistèmes du Centre National de la Recherche Scientifique (LAAS/CNRS) da cidade de Toulouse (França) pelo apoio material, intelectual e tecnológico, que tornou possível a finalização deste trabalho.

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

O Planejamento do "Start-Up" de Plantas Químicas através de Redes de Petri e Algoritmos Genéticos

Márcia Aparecida Fernandes

Outubro, 1996

Orientador: Prof. Luís Alfredo Vidal de Caravalho

Co-Orientador: Prof. Felix Mora-Camino

Programa: Engenharia de Sistemas e Computação

A síntese de procedimentos operacionais para indústria de processos tem sido estudada nos últimos anos tanto por pesquisadores em Teoria de Controle quanto por pesquisadores em Inteligência Artificial. Acredita-se que as duas principais dificuldades envolvidas neste problema são a escolha de uma representação adequada para o processo e a síntese de um eficiente método de busca. Então, neste trabalho estas duas dificuldades são tratadas. As representações da planta e do processo propostas usam redes de Petri híbridas de maneira que as atividades discretas e contínuas são convenientemente modeladas. Assim, Algoritmos Genéticos e técnicas de Computação Evolucionária são usados para busca de uma sequência ótima de operações que realiza o processo de inicialização da planta.

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

The Planning of Chemical Plants Start-Up By Petri Nets and Genetic Algorithms

Márcia Aparecida Fernandes

October, 1996

Thesis Advisor: Prof. Luís Alfredo Vidal de Carvalho

Thesis Co-Advisor: Prof Felix Mora-Camino

Department: Systems Engineering and Computer Science

The synthesis of operational procedures for process industry have been studied in the last years by control engineers as well as by Artificial Intelligence researchers. It appears that the two principal difficulties attached to this problem are the choice of an adequate representation for the process and the synthesis of an efficient search method. So, in this work these two difficulties are coped with. The plant and process representations proposed here use an Hybrid Petri Net formalism so that its continuous and discrete activities are conveniently modeled. Then, Genetic Algorithms and Evolutionary Computing techniques can be used to search for an optimal sequence of operations which starts-up the plant.

Índice

Capítulo 1

Introdução

Neste trabalho, considera-se o problema da síntese de procedimentos operacionais para o "start-up" (partida) de plantas químicas. Este problema pode ser visto em diferentes circunstâncias tais como: durante a concepção dos processos químicos, em procedimentos de manutenção de uma planta química, ou ainda, quando o processo atual deve ser parado e outro deve ser inicializado na planta. Observe que, neste último caso também se faz necessário a síntese de procedimentos operacionais para parar ("stop-down") um processo.[1]

A princípio, o problema da determinação e da execução dos procedimentos operacionais para o "start-up" de uma planta química era resolvido levando em consideração pricipalmente, o operador humano. Este tanto determinava, através de sua experiência os procedimentos operacionais necessários quanto os executava. Entretanto, com o avanço tecnológico e industrial, a determinação destes procedimentos tornou-se complexa, desde que envolve várias restrições relativas ao processo e aos equipamentos. E a execução tornou-se de alto risco para o operador humano devido as substâncias manipuladas durante o processo e

[1]Embora, o "start-up" e o "stop-down" tenham características semelhantes, este trabalho se restringe ao tratamento do problema do "start-up".

também aos equipamentos. De fato tais riscos ocorriam mesmo em plantas menos complexas, o avanço tecnológico apenas agrava tal problema.

A respeito do problema do "start-up", pode-se observar também, que devido á presença de algumas características este problema foi identificado como um problema quantitativo, dado que têm suas origens em áreas como a Engenharia. Entretanto, outras características, ditas simbólicas, levaram alguns pesquisadores desta área como por exemplo, Rivas et al.[2, 3], Foulkes et al.[5] e Fusillo et al.[10, 11] a procurar técnicas diferentes das propostas nestas teorias. De fato, tal problema foi visto como o problema de planejar planos de ações, ou seja, a síntese de procedimentos operacionais é na verdade a determinação de um plano de ações que quando executado realiza o "start-up" de um processo ou de uma planta química.

Devido a estas características simbólicas, as técnicas, modelos e teorias da área de Inteligência Artificial foram introduzidas na pesquisa deste problema, visando a nova forma de solução necessária. Muitos dos tópicos desenvolvidos ou originários nesta área, tais como, Sistemas Especialistas e Planejamento, têm sido utilizados na tentativa de resolver questões como, por exemplo, detecção de falhas, tomadas de decisão e determinação de planos de ação envolvidas diretamente na solução deste problema [28, 29, 30, 36] .

Além de Sistemas Especialistas e Planejamento, também Algoritmos Genéticos[14, 18] têm sido usados na solução de alguns tipos de problema de Engenharia, como pode ser visto em Grefenstette[15] e em Goldberg[19, 20, 21], que os utilizou especialmente para a otimização de parâmetros de controle. Koza em [26, 27] apresenta uma proposta geral para a determinação da sequência de ações através de Programação Genética. Também os

Sistemas Classificadores[22] são uma tentativa de utilização de Algoritmos Genéticos para a aprendizagem de regras.

Assim, o principal objetivo deste trabalho é desenvolver uma nova abordagem de otimização para o planejamento do "start-up" de plantas químicas complexas. A proposta é uma aplicação de Algoritmos Genéticos à questão da determinação de sequências (planos) de ações para o "start-up" de plantas químicas. Alguns aspectos deste problema foram tratados apenas pelas teorias de controle como é argumentado em [4, 2], mas as teorias e técnicas encontrados em Planejamento foram adotados por alguns pesquisadores[12, 13], dado que nesta subárea de Inteligência Artificial o principal objetivo é a criação de planos de ações.

Assim, o que se pretende é construir um plano de ação para o "start-up" de plantas químicas utilizando Algoritmos Genéticos, ou seja, planejar através de Algoritmos Genéticos o "start-up" de plantas químicas. Ou ainda, de outra forma mais geral realizar Síntese de Procedimentos Operacionais através de algoritmos genéticos e técnicas de Planejamento. Mas, como leva-se em consideração a natureza híbrida do sistema, isto é, a presença de características discretas e contínuas do processo, fez-se uso de uma ferramenta de representação capaz de acolher simultaneamente tais particularidades deste tipo de problema, as Redes de Petri Híbridas[32]. Resumindo, é uma tentativa de tratar tal problema ao nível de sua complexidade, através de Algoritmos Genéticos e tendo como base aspectos básicos de teorias de Planejamento.[2]

No capítulo 2 apresenta-se o problema discutindo a sua complexidade, a necessidade de

[2]Vale dizer que apesar de se tratar do "start-up" de plantas químicas poder-se-ia considerar outros tipos de plantas como as elétricas, ou até mesmo sistema que não necessariamente sejam vistos como plantas.

resolvê-lo, as dificuldades quanto à sua representação e a influência desta nos métodos de busca pela solução. Discute-se ainda o uso dos métodos da Teoria de Controle para a sua resolução, dado que é um problema com origens na área de Engenharia. Mostra ainda a relação deste problema com o problema de supervisão. E por fim a introdução e a influência das técnicas da área de Inteligência Artificial nas propostas de resolução atual.

No capítulo 3 discute-se alguns modelos que enfatizam o uso de Inteligência Artificial para a solução do problema do "start-up" de plantas químicas. O capítulo 4 descreve uma estrutura para representação, Redes de Petri, que é capaz de abranger diferentes tipos de conhecimento envolvidos no problema. E além disso, é também capaz de modelar a natureza híbrida (aspectos contínuos e discretos) do problema.

Uma descrição geral dos Algoritmos Genéticos, suas aplicações e uma análise das novas versões introduzidas são apresentadas no capítulo 5. Finalmente, o capítulo 6 mostra o desenvolvimento da pesquisa que se faz neste trabalho juntamente com exemplo e resultados e no capítulo 7 tem-se as conclusões e perspectivas futuras.

Capítulo 2

Apresentação do Problema

A necessidade de redução de custos, de níveis de riscos e de aumento de eficiência em geral nos setores industriais motivou o crescimento das pesquisas em diferentes áreas do conhecimento no sentido de criar modelos para a automação dos processos industriais. Tão grande foi e tem sido tal necessidade que atualmente praticamente todas as etapas de criação de um processo vêm sendo automatizadas. A primeira destas etapas é aquela em que se faz a avaliação econômica, comercial e tecnológica do processo. Nas duas etapas seguintes, projeto e análise, tem-se a realização propriamente dita do processo.

A etapa de projeto envolve a preparação de todos os recursos a serem utilizados, como equipamentos, propriedades físicas e diagrama de fluxo, cujo término se dá com a construção física de uma planta. As duas primeiras etapas constituem a síntese do processo. A última fase, a análise, tem como objetivo melhorar ou ajustar estratégias operacionais e/ou modificação de recursos físicos utilizados, a partir da observação da planta em operação Fair[1].

A atividade de interesse neste estudo é aquela que ocorre após a construção da planta, denominada inicialização ("start-up") da planta. Neste momento deve-se realizar a síntese dos procedimentos operacionais necessários para esta inicialização, ou seja, deve-se verificar

e estabelecer as condições necessárias afim de que a planta possa ser operacionalizada para realizar o processo planejado. Sendo assim, o objetivo é a determinação dos passos a serem seguidos para colocar a planta em operação.

Observe que diferentes sínteses podem ser identificadas na construção de processos. Por exemplo, tem-se a síntese do processo que consiste em duas etapas: a avaliação dos aspectos econômicos, tecnológicos, comerciais entre outros, e projeto, que é a concepção do processo. Após esta síntese, há a síntese dos procedimentos para a inicialização do processo e da planta. Assim, uma descrição esquemática da construção de um processo pode ser como na figura 2.1. Entretanto, vale lembrar que há constante interação entre todas estas etapas.

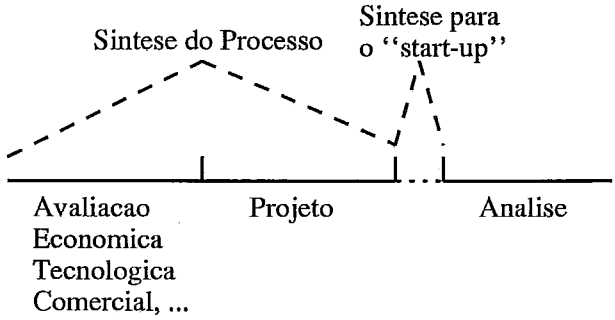

Figura 2.1: Etapas para construção de um processo

Pode-se observar que quando se trata de plantas complexas, a síntese de procedimentos operacionais terá um alto nível de complexidade. De fato, um grande número de restrições de diferentes tipos devem ser levadas em consideração, como por exemplo, os modos e os estados de operação de cada equipamento, as restrições de segurança e também aquelas relacionadas aos resultados pretendidos com a execução do processo. A princípio (e ainda hoje), esta era uma atividade realizada por operadores humanos. Entretanto, com o crescimento industrial

e tecnológico tornou-se bastante complexa e de alto risco econômico e humano, por isso a síntese automática, ou ainda, auxiliada por computador, pode evitar graves riscos e prejuízos.

Considere, por exemplo, a concepção de um processo químico: A síntese de procedimentos operacionais para o "start-up" da planta química (ou parte da planta) relativa ao processo pode envolver restrições como evitar a mistura de substâncias, observar se a operação dos equipamentos ocorre de acordo com os objetivos pretendidos, ou ainda, se está de acordo com as normas da própria construção do equipamento, observar temperaturas, quantidades físicas das substâncias e taxas de fluxo, entre outras. Além disso, pode haver um alto grau de dependência entre diferentes estágios do processo, isto é, podem existir dependências entre as restrições o que favorece a propagação destas.

A presença de restrições que envolvem a determinação e a observação das variações de parâmetros quantitativos, como por exemplo, temperaturas e taxas de fluxo, fez com que os métodos que pertencem à teoria de controle tenham sido bastante utilizados, dado que estes são parâmetros de controle. Na verdade, a determinação dos parâmetros quantitativos é um aspecto relevante para as propostas de solução do problema. Mas, este é apenas um dos aspectos da resolução do problema, desde que existem outras restrições, como por exemplo, verificar modos de operação dos equipamentos que não são apenas quantitativas, mas sim qualitativas, exigindo assim, um tratamento simbólico. Além disso, a solução desejada pode ser vista como uma sequência de passos que colocam a planta em operação e por isso há mais do que questões numéricas. Vale dizer que o fato deste problema ter tido suas origens na área de engenharia fez com que os pesquisadores (engenheiros essencialmente) o enquadrassem

como um problema tipicamente quantitativo.

Assim, o que se observa é a existência de dois subproblemas: a parte numérica em que é necessário a determinação de parâmetros de controle do processo; e a parte simbólica que envolve a determinação de procedimentos ou ações, nos quais podem estar presentes ou não os parâmetros numéricos. Note, entretanto, que estes dois subproblemas estão intrinsicamente ligados.

A partir disto, os pesquisadores do problema de síntese de procedimentos operacionais procuraram em outras áreas do conhecimento, teorias e ferramentas que pudessem tratar a parte simbólica do problema. E uma área que efetivamente contribuiu e tem contribuído neste aspecto é a Inteligência Artificial, dado que sua característica básica é o tratamento de questões simbólicas relacionadas com problemas em geral.

Sob o aspecto geral, pode-se dizer que as pesquisas desta área se dividem em dois pontos básicos, representação do conhecimento e métodos de busca. E de fato, estes são pontos importantes para qualquer tipo de técnica para resolução de algum problema. Além disso, há argumentos que afirmam que quanto melhor a estrutura de representação, mais eficientes serão os métodos de busca.

No caso específico do problema de Síntese de Procedimentos Operacionais existe uma diversidade de tipo e quantidade de conhecimento a ser representada, como por exemplo, os equipamentos (unidades operacionais) e as quantidades físicas envolvidos no processo, o próprio processo, a planta, os estados da planta ou processo, e as ações que podem ser as intervenções sobre os equipamentos. E por isso, dificuldades consideráveis são encontradas

ao se tentar adotar uma forma de representação que possa minimizar a complexidade dos métodos de busca. E este é um dos motivos que faz com que as pesquisas desenvolvidas em Inteligência Artificial desempenhem um papel importante na resolução deste problema.

Especificamente, as ferramentas de um tópico desta área, denominado Planejamento em Inteligência Artificial, têm sido utilizadas. Através de sua definição pode-se entender porque foi introduzido. De acordo com Gentil[36], é a busca de uma sequência de ações que permitem atingir um fim a partir de um dado estado inicial. Ou seja, um método de busca tenta construir a partir de um dado estado do processo (estado inicial), utilizando a estrutura de representação, uma sequência de ações e/ou estados que conduza o processo ao estado desejado (estado final ou meta). A esta sequência de ações dá-se o nome de plano. Para Smuts et al.[30], Planejamento pode ser visto como uma simulação do mundo real através da manipulação de símbolos para investigar as ações esperadas sem ter de executá-las. Observe que estas definições vêm de encontro aos objetivos pretendidos para a solução do problema de síntese.

Segundo Smuts et al.[30], as técnicas de Planejamento são ferramentas discretas, isto é, estão baseadas em representações do conhecimento discretas. E por isso, deve-se procurar regras que possam mapear o mundo real contínuo em representações discretas, dado que estas técnicas de Planejamento são simples se comparadas às necessidades do mundo real. Na verdade, nas atividades de síntese estão envolvidas ações contínuas para as quais os símbolos podem não ser adequados, pois estes em geral têm características discretas.

Também os Sistemas Especialistas em Inteligência Artificial têm sido utilizados em pro-

blemas que envolvem Síntese de Procedimentos Operacionais, cujo papel neste caso é o de extrair informações a partir da análise do estado do processo. Tais informações são utilizadas para direcionar tomadas de decisão e estas especificam as ações que devem ser executadas visando a garantir o controle do processo. Sendo assim, nesta parte podem estar envolvidas por exemplo, técnicas para tomadas de decisão, para construção de sequência de ações e para aprendizagem de comportamento entre outras. Um método assim composto está mais relacionado à atividade de supervisão do processo durante toda a sua realização. E neste caso a síntese seria vista apenas como parte da atividade de supervisão. Na verdade, Sistemas Especialistas seriam a forma de passar informações entre os diversos módulos de um sistema para supervisão e controle como o da figura 2.2.

Outros tópicos que são originários ou foram desenvolvidos na área de Inteligência Artificial também podem ser citados como fontes de ferramentas aplicáveis à solução do problema de síntese, como por exemplo, Redes Neuronais e Algoritmos Genéticos[28, 29, 30].

2.1 Visão Geral das Técnicas de Síntese para o "Start-up"

Nas primeiras técnicas[2, 3, 4] para a Síntese de Procedimentos Operacionais, muitas das questões mencionadas anteriormente não chegaram a ser tratadas. Na verdade, eles estavam envolvidos em técnicas mais simples, devido às restrições computacionais da época. Por isso, uma relevante simplificação feita foi tomar todos os equipamentos sob o ponto de vista de um único equipamento, a válvula, e a planta original foi transformada em outra que continha apenas válvulas e tubulações. As ações eram as manipulações de abrir ou fechar válvulas, já

que não existiam outros equipamentos. Isto foi uma tentativa inicial de utilizar as idéias de Planejamento, mas tentando minimizar as dificuldades do aspecto representacional do problema. Assim, a solução era encontrada através de algum método, geralmente exaustivo, que realizava a busca pela sequência adequada de operações de válvulas, considerando apenas restrições de segurança.

No entanto, esta simplificação do problema de síntese de procedimentos operacionais não significava uma eliminação completa das dificuldades, dado que a quantidade de válvulas e consequentemente suas combinações continuavam sendo fatores de complicação. A simplificação estava relacionada à possibilidade de não se considerar determinadas restrições relativas a equipamentos mais sofisticados que a válvula, pois este estava subjacente à operação da válvula.

Os trabalhos que se seguiram, como o de Fusillo et al.[10], foram direcionados pela real complexidade do problema, e assim consideravam não só restrições de segurança, mas também aquelas relacionadas à natureza (química, física, ...) e aos equipamentos do processo. E, desta maneira, se estabeleceu uma outra corrente de pesquisa, que embora fosse uma evolução da anterior, não a anulava completamente. Isto porque, apesar de usarem metodologias mais elaboradas, alguns trabalhos ainda apresentavam soluções que se resumiam a seqüências de manuseio de válvulas.

De acordo com Fusillo et al.[10], estas são as duas classes relevantes dos trabalhos até então produzidos. Isto é, ou a síntese se resume em apresentar uma sequência de situações de válvulas que respeite uma única restrição, a segurança local; ou é vista como a geração

de sequências de transição de estado do sistema, baseada em modelos de mais alto nível.

Sendo assim, o desenvolvimento de procedimentos de resolução do problema de síntese depende da capacidade da estrutura de representação em acolher e associar diferentes tipos de conhecimento e de um algoritmo capaz de realizar de forma eficiente uma busca de informações através desta representação. Por isso, a evolução dos modelos foi marcada, em parte, pela introdução de teorias que se encontravam na área de Inteligência Artificial.

2.2 Diferenças entre Síntese de Procedimentos Operacionais e Supervisão

O problema que se pretende abordar neste trabalho é a Síntese de Procedimentos Operacionais para o "start-up" de plantas químicas. Como visto acima, este é um problema que envolve aspectos quantitativos e qualitativos. Por isso, vale ressaltar que devido aos aspectos qualitativos, as ferramentas e teorias da área de Inteligência Artificial foram e são utilizadas. Mas, um outro fator a considerar é que mesmo havendo um aspecto compatível com o uso de Inteligência Artificial, o ponto de vista de pesquisadores desta área em relação a algum problema não é idêntico àquele que tem os pesquisadores da área de origem do problema. A síntese de procedimentos operacionais é um exemplo para esta argumentação.

De uma certa forma, os pesquisadores de áreas como a Engenharia olham as teorias e as formas de resolução de problemas propostas em Inteligência Artificial como algo com pouca base realmente formal (matemática) apesar das origens lógicas desta disciplina. Por exemplo, os dois aspectos do problema acima podem ainda serem ditos contínuos (quantitativos) e discretos (qualitativos). E de acordo com Smuts et al.[30] as ferramentas desta área são

discretas. Disto pode-se argumentar que os aspectos contínuos deste problema, em geral não são acompanhados por técnicas como as encontradas em Planejamento, por exemplo. Entretanto, isto não quer dizer que as propostas em Planejamento sejam fracas, mas sim que elas tratam especificamente de um aspecto do problema, a saber, aquele que é compatível com os objetivos desta área. Talvez, uma questão a ser determinada antes do emprego de alguma ferramenta de Inteligência Artificial seja "Onde utilizá-la e até que ponto?" Apenas uma boa resposta para esta questão permitirá uma análise justa das expectativas e resultados em relação à Inteligência Artificial.

Um exemplo desta análise sobre onde aplicar Inteligência Artificial pode ser vista em Smuts et al.[30]. Para este, as ferramentas utilizadas por engenheiros para projetar sistemas de controle não são adequadas para sistemas de supervisão. Considera que a supervisão seja um procedimento que utiliza conhecimento aplicável e fatos para decidir quais ações dar a um controlador automático para que o processo realize certas metas (figura 2.2). Acrescenta que o supervisor apresenta funções lógicas (qualitativas) enquanto o controlador automático apresenta funções numéricas (quantitativas), e por isso são inerentemente diferentes exigindo assim diferentes métodos de tratamento.

Baseado nestas idéias, ele considera que o supervisor é constituído das atividades de geração de plano, execução de plano, monitoração de execução e manuseio de exceções, que foram identificadas a partir da observação de supervisores de controle humanos (veja figura 2.3).

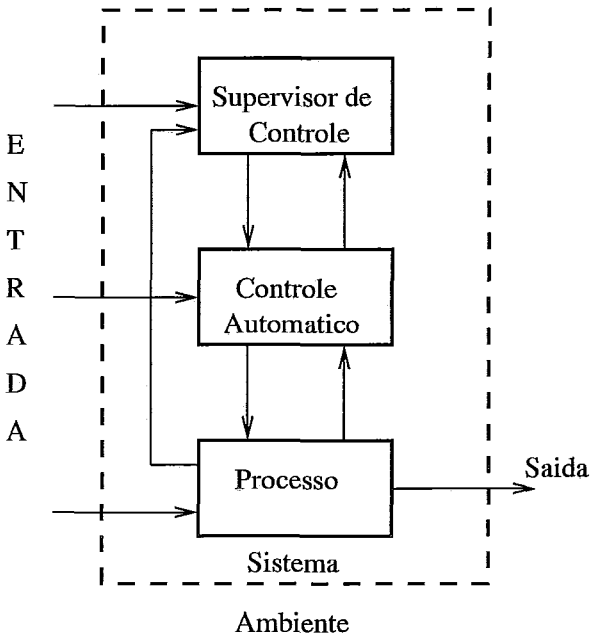

Figura 2.2: Sistema de Controle e Supervisão

O que pode se observar nesta abordagem[1] é que o problema a ser resolvido por técnicas de Inteligência Artificial envolve um tipo de conhecimento humano, que é tanto difícil obter quanto de formalizar. De fato, o conhecimento do supervisor humano, não necessariamente é o mesmo conhecimento dos engenheiros que projetaram e construíram o processo. Na maioria das vezes, os supervisores humanos se baseiam em experiência e não em conceitos matemáticos para executarem determinadas ações. E sobre este ponto de vista, as ferramentas desta área têm tido sua aplicabilidade verificada.

A característica acima é o principal argumento dos pesquisadores do problema para introduzirem técnicas de Inteligência Artificial em suas propostas para resolução. Na verdade, pode-se identificar uma relação entre estes dois problemas, supervisão e síntese, parte da

[1]Também utiliza técnicas de Planejamento.

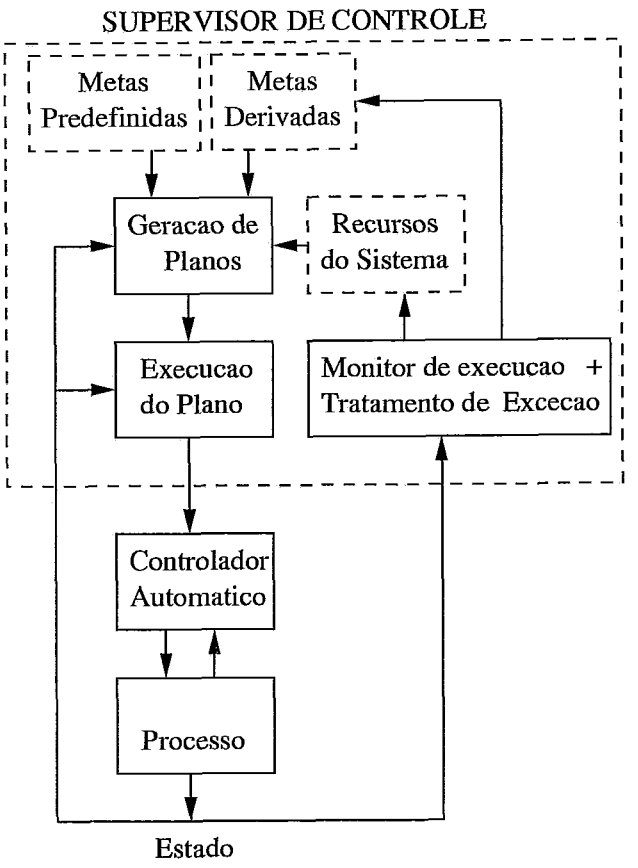

Figura 2.3: Modelo de Supervisão

solução do primeiro é também a solução para o segundo, isto é, o planejamento de ações. Disto pode-se constatar que o problema de Síntese de Procedimentos Operacionais pode estar envolvido num problema maior que é o da supervisão, dado que a síntese pode ser necessária em várias fases da evolução de um processo. No entanto, a supervisão é um problema mais geral envolvendo observação e reavaliação permanente da evolução do processo, podendo então estar presente na etapa de análise do processo. E por isso, inclui outros módulos funcionais além do de planejamento. Enquanto a síntese de procedimentos operacionais para o "start-up" está dedicada a um momento específico da construção do processo.

Capítulo 3

Análise de Algumas Técnicas

A apresentação e análise das técnicas que se seguem se baseiam nos aspectos de representação do conhecimento, método de busca e o emprego das técnicas de Planejamento. Como argumentado anteriormente estes aspectos podem estabelecer diferenças relevantes entre as diversas técnicas para síntese de procedimentos operacionais. Assim, usa-se uma forma de apresentação esquemática para ressaltar as principais características dos modelos em relação a tais aspectos.

Por exemplo, em relação a Planejamento, apresenta-se as representações utilizadas para estados do sistema (em particular, para os estados inicial e final) e para o plano de ação resultante. Além disso, identifica-se as restrições que foram consideradas e o(s) método(s) de busca empregado(s)

Os dois primeiros modelos que se seguem são exemplos da primeira corrente, ou seja, a reduzem os equipamentos a um único equipamento, a válvula. Apesar de serem limitados para tratar plantas de interesse industrial, introduziram idéias seguidas em modelos posteriores.

Modelo de Rivas

Em Rivas et al.[3] tem-se um dos primeiros modelos para síntese de procedimentos ope-

racionais. A figura 3.1 mostra uma planta química composta de válvulas e recipientes, que é uma transformação da planta original.

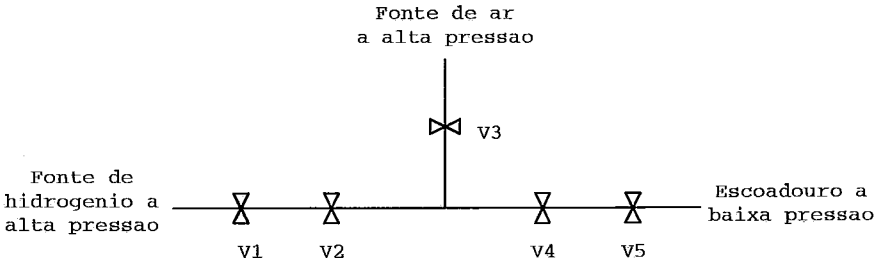

Figura 3.1: Planta para o modelo de Rivas

Considere o seguinte exemplo:

- **Estado inicial:** presença de ar.
- **Estado final:** fluxo de hidrogênio a baixa pressão.
- **Observação:** Tais estados podem ser representados em termos de situações das válvulas como na figura 3.2.
- **Restrições Consideradas:** Evitar o contato entre hidrogênio e ar (de segurança).
- **Representação do Conhecimento:** Proposições lógicas específicas que descrevem se uma substância flui ou não e em qual extremidade do conector isto ocorre.
- **Método de Busca:**
  - toma-se as diferenças (figura 3.2) que separam os dois estados, que são os estados operacionais das válvulas $V1$, $V3$ e $V5$.

- Transforma-se a meta "Fluxo de hidrogênio a baixa pressão" nas submetas "Evacuar ar" e "Inicialize o fluxo de hidrogênio". "Evacuar ar" em "Pare o fluxo de ar" e "Abra sistema a baixa pressão de escoamento".

- Identifica-se "Pare o fluxo de ar" com fechar $V3$, "Abra sistema a baixa pressão de escoamento" com abra $V5$ e "Inicialize o fluxo de hidrogênio"com abrir $V1$.

- **Plano ou Sequência Final:** fechar V3, abrir V5 e abrir V1.

	V1	V2	V3	V4	V5
Estado Inicial	Fechada	Aberta	Aberta	Aberta	Fechada
Estado Final	Aberta	Aberta	Fechada	Aberta	Aberta
Diferencas	X		X		X

Figura 3.2: Tabela para o exemplo de Rivas

A partir deste exemplo pode-se observar algumas características encontradas nas técnicas de Planejamento e que fizeram parte dos primeiros geradores de plano desenvolvidos, como por exemplo, o STRIPS[7]: a determinação das diferenças entre os dois estados e a construção de submetas de forma hierárquica.

Segundo O'Shima[4], os algoritmos da primeira corrente são procedimentos compostos de duas partes. A primeira parte é uma busca por uma rota particular para o fluxo entre pontos inicial e terminal especificados e a segunda consta de uma avaliação da situação dos equipamentos nesta rota. Um exemplo de tal algoritmo é o modelo de Foulkes et al.[5], onde a intenção básica era investigar um sistema que tivesse conhecimento da configuração da planta, por isso utilizou a planta original como na figura 3.3, que contém apenas válvulas,

bombas e recipientes.

Modelo de Foulkes

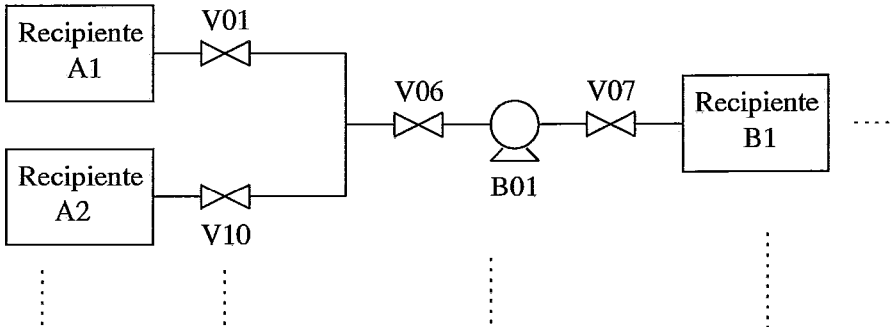

Figura 3.3: Planta para o modelo de Foulkes

- **Estado Inicial:** Ponto da planta onde inicia o fluxo do processo. Ex.: $A1$.
- **Estado Final:** Ponto da planta onde termina o fluxo do processo. Ex.: $B1$.
- **Restrições Consideradas:** operacionais (equipamentos), de segurança e as relativas às mudanças necessárias no estado atual da planta para estabelecer o estado desejado.
- **Representação do Conhecimento:**
  - Fragmentos da planta tendo as válvulas ou algum recipiente como extremidades.
  - Junta-se a isto uma lista contendo as situações das válvulas que devem estar abertas ou fechadas para permitir fluxo através deste fragmento.
  - A presença de equipamentos no fragmento é adicionada a esta lista.
  - Exemplos:
    1. **link(01,06): [V01(aberta), V06(aberta), V10(fechada)]**

2. **link(06,07): [V06(aberta), V07(aberta), B01(ligada)]**

3. **link(07,B1): [V07(aberta)]**

4. **Observação:** Não há descrição da topologia da planta.

- Uma rota completa seria uma sequência destes fragmentos.

- **Métodos de Busca:**

  - Passo 1: Do ponto inicial, $p_o = A1$, busca-se um fragmento que o ligue ao ponto final ou a algum ponto intermediário.

  - Passo 2:

    Se ponto intermediário, p_i, é atingido

    então Passo 1 com $p_o = p_i$

    caso contrário

    se ponto final é atingido

    então Passo 3

    caso contrário falha

  - Passo 3: Verificar se as condições da sequência de fragmentos são válidas. Suponha V10(aberta), V06(fechada) e B01(desligada).

- **Plano ou Sequência de Ações:** É constituído das condições não verificadas no Passo 3, V10(fechada), V06(aberta) e B01(ligada), observando as restrições de inicializar bomba após todas as válvulas a ela relacionada estarem abertas.

Um ponto a se observar neste modelo é o uso da planta original, embora as rotas não sejam geradas diretamente da planta, dada que é feita a partir dos fragmentos. Além disso, também considera equipamentos diferentes da válvula. Assim, a contribuição deste modelo é a introdução de uma representação mais elaborada, embora baseada no modelo de Rivas et alli[2].

Os modelos seguintes fazem uso de representações e plantas mais complexas, adotam técnicas de Planejamento mais elaboradas e consideram equipamentos diferentes da válvula. Sendo assim, fazem parte da segunda corrente das pesquisas em síntese de procedimentos operacionais. No modelo de Tomita et alli[6] por exemplo, utiliza-se regras sintáticas, por isso é possível representar vários tipos de equipamentos.

Modelo de Tomita

- **Representação do Conhecimento: gramática LISP**

1. Dados Estáticos: Planta e Equipamentos:
   - Grafo dirigido: nós (equipamentos, exceto as válvulas) e arcos (tubulação).
   - Modelagem funcional: cada nó ou arco é como no quadro 3.1(a) e (b)
   - Principais Características do Nó:
     * Caminho Aberto: caminho onde o fluxo do processo é possível.
     * Condição de caminho aberto: Contém as restrições relativas à operação do nó e que devem ser satisfeitas antes da operação.
     * Lista_Condicao_Caminho_Aberto é NIL, se o nó não tem restrições ou

```
<No> ::= ( <Nome><Estado><Lista-de-Arcos>
           <Lista-de-Trabalho-Preparatorio>
           <Lista-de-Regras-Funcionais>
           <Lista-de-Condicoes-de-Caminho-Aberto> )

<Nome> ::=permutador-de-calor |
           condensador

<Estado> ::= ligado | desligado

<Lista-de-Arcos> ::= ( <Lista-de-Arcos-Entrada>
                       <Lista-de-Arcos-Saida> )

<Lista-de-Regras-Funcionais> ::= ( <Regras-Funcionais>
                                   <Lista-de-Regras-Funcionais> )
<Regra_Funcional> ::=( IF <parte-if>
                       THEN <parte-then>
                       ARCO <Lista-de-Arcos-Correspondentes> )

                            (a)

<Arco> ::= (<Estado><Tipo-de-Fluxo><Temperatura><Pressao>
           <Lista-Componentes-do-Fluxo> )

<Estado> ::= ( AND <Estado-Valvulas> ) | T

<Tipo-de-Fluxo> ::=  GAS | LIQ | MISTURA

<Lista-de-Componentes-de-Fluxo> ::= ( <Componentes> )

                            (b)
```

Quadro 3.1: Representações do modelo de Tomita et al.

não está em um caminho aberto; ou é uma lista de caminhos abertos que resolvem as restrições no nó.

- Principais Características do Arco:
  * Estado do arco: AND lógico dos estados das válvulas.
  * Válvula: componente de algum arco.
- Regras funcionais:
  * especificam os diferentes modos de operação dos equipamentos.
  * identificam os arcos ocupados e livres.
  * evitam contatos indevidos (restrições de segurança.)
  * Parte IF: estados dos arcos de entrada.
  * Parte THEN: como os estados dos arcos de saída serão alterados.
  * Parte ARCO: uma coleção de listas relacionando arcos de entrada com arcos de saída.
- Recursos do Nó:
  * Especificam restrições particulares dos equipamentos.
  * Exemplos: verificação do nível de óleo e das válvulas de segurança, controle de aquecimento e resfriamento.
- **Observação:** Topologia da planta pode ser extraída dos nós a partir das listas de arcos.

2. Dados Instantâneos: Desenvolvimento da Síntese:

- Conhecimento *Fragmentário*: conjunto de procedimentos PROLOG, que constitui uma máquina de inferência para verificar condições e restrições momentâneas.
  * Exemplos: verificar o uso de um equipamento, existência de fluxo, condições especiais de operação e associar estados da planta.
- **Observação:** Este conhecimento é dito heurístico pois auxilia na busca e determinação de estados intermediários da planta.

- **Estado Inicial:** Especificado pelos arcos considerados início do processo, ou seja, os arcos de entrada para o processo.

- **Estado Final:** Arcos de saída para o processo.

- **Restrições Consideradas:** Estão incluídas nas regras funcionais (gerais), nos recursos do nó (específicas) e nos trabalhos preparatórios.

- **Observação:** Dado que a modelagem é principalmente funcional, não se tem restrições numéricas, como por exemplo, exigências de produção e valores das variáveis físicas.

- **Métodos de Busca:**

- Utiliza duas formas de busca associadas aos dois tipos de conhecimento (figura 3.4):

1. Busca em Retrocesso utilizando o conhecimento estático: Cria submetas a partir dos arcos de saída, procurando por uma regra funcional que tenha estes arcos na

sua parte THEN. Cria-se um nó ligando os arcos de entrada da parte IF com este arco de saída.

- Exemplo: "A" é o arco de entrada, E e H são os arcos de saída. O nó 6 é criado para representar um possível equipamento que realizará a ligação entre G e H. Toma-se G e cria-se um novo nó, e assim até A.

2. Busca Progressiva utilizando o conhecimento fragmentário:

   - Tendo-se a lista de equipamentos inativos na planta
   - Passo 1: toma-se um equipamento X.
   - Passo 2:

     Se X não está em algum caminho aberto

     então

     Se X tem condições de caminho aberto

     então Passo 1 para resolvê-las

     Se Passo 1 não resolve

     então Máquina de Inferência

     caso contrário Passo 1

     caso contrário

     trabalhos preparatórios,

     recursos do nó,

     simulação das regras funcionais,

     cria-se novo estado, ou seja, novo(s) arco(s) de entrada,

Passo 1

- **Observação:** A máquina de inferência tenta encontrar um equipamento que possa ser operacionalizado no estado atual da planta.
- Exemplo: Busca pelos equipamentos (submetas) 1, 2, 3, 4, 5 e 6 que trasformem A em B, B em C, C em D e F, D em E, F em G e G em H.

- **Plano ou Sequência de Ações:** Construída no Passo 2 de maneira ordenada.

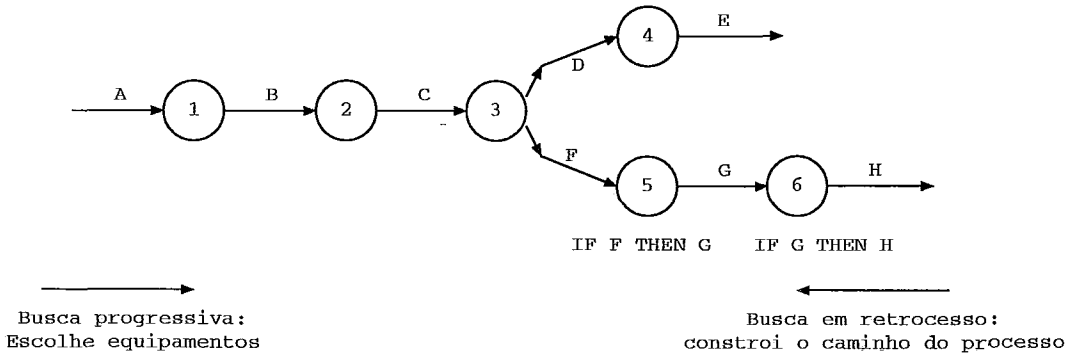

Figura 3.4: Exemplo para o modelo de Tomita eet al.

Este modelo não utiliza técnicas específicas de Planejamento, na verdade explora ferramentas mais gerais, como as linguagens LISP e PROLOG, que têm grande capacidade para expressar conhecimento simbólico e para construir métodos de raciocínio como máquinas de inferência. E a justificativa para estas escolhas foi o objetivo de incluir no modelo o conhecimento e a experiência de operadores humanos, que é essencial mesmo em sínteses auxiliadas por computador.

Modelos de Fusillo

As técnicas de Fusillo et al[10, 11] introduzem informação quantitativa (equações numéricas

que descrevem os equipamentos), ignoradas no modelo anterior. Além disso, são exemplos de uso efetivo das teorias e técnicas de Planejamento de mais alto nível, como por exemplo, as teorias introduzida pelos geradores de planos STRIPS[7], NOAH[8] (Redes Procedimentais) e MOLGEN[9] (Planejamento com restrições), que estão relacionados entre si de forma evolutiva.

1. **Primeiro Modelo**

   - **Representação do Conhecimento:**

```
INICIAR_AQUECEDOR(Valor)
     TEMP  <---  Valor
     PRESSAO  <---  fcn(Valor)

 onde, "Valor" representa a nova temperatura e
"fcn", uma funcao que avalia o novo valor da
pressao
```

(a)

ESTADO INICIAL
TEMP = baixa
PRESSAO = baixa
Fluxo = 0
Fase = gas
[HCl] = alto
[CH4] = baixo

(b)

Figura 3.5: Representação do primeiro modelo de Fusillo et al.

– Planta

* Estados da planta: vetor de variáveis de estado (figura 3.5(b)).

* Decomposição da planta em subsistemas compostos de um ou mais equipamentos e que contenham *estado estacionário.*

* Estado Estacionário: situação em que a meta operacional é atingida ou se tem um estado estável da planta ou um estado em que as alterações são lentas.

* **Observação:** Não há representação para a topologia da Planta.

- Equipamentos

* Fontes ou sumidouros de energia, momento ou material.

* Rede IS-A: descreve as relações de herança entre os diversos atributos dos equipamentos.

* Exemplo: um nó do primeiro nível pode ser um permutador de calor descrito como uma fonte ou sumidouro de energia. No nível abaixo, pode-se ter um aquecedor, como um tipo de fonte, e um resfriador, como um sumidouro de energia.

- Ações

* Descrevem as variações dos valores das variárveis em relação às operações dos equipamentos (figura 3.5(a)).

- **Métodos de Busca:** Busca pelos estados estáveis dos subsistemas (supondo desprezíveis as interações) através da estratégia meio-fins[7]

- Compara os estados inicial e final para estabelecer as diferenças entre os valores das variáveis e as alterações necessárias.

- Identifica uma meta com a diferença e propõe-se um operador para realizá-la.
- A partir da rede IS-A seleciona os equipamentos que podem realizar a meta.
- Exemplo:
  * Diferença na temperatura e a alteração é no sentido de aumentar.
  * Meta associada: "aumentar a temperatura do sistema de baixa para alta".
  * Tendo o equipamento "X" que a realiza a meta, o operador seria "aumentar o nível de operação de X".
- Ordenação dos operadores: utiliza a metodologia da rede procedimental de Sacerdoti[8], que faz ordenações observando restrições.
- Simulação dos efeitos dos operadores.

2. **Segundo Modelo**

- **Representação do Conhecimento: Modelo Causal**
  - Dígrafos:
    * descreve cada equipamento a partir das relações entre as variáveis físicas que este manipula.
    * Considere o exemplo do permutador de calor (figura 3.6), as variáveis são temperatura e taxa de fluxo. Os dígrafos são como na figura 3.7.
    * O valor GANHO associado a cada arco indica a direção causa- efeito entre as variáveis e é determinado por:

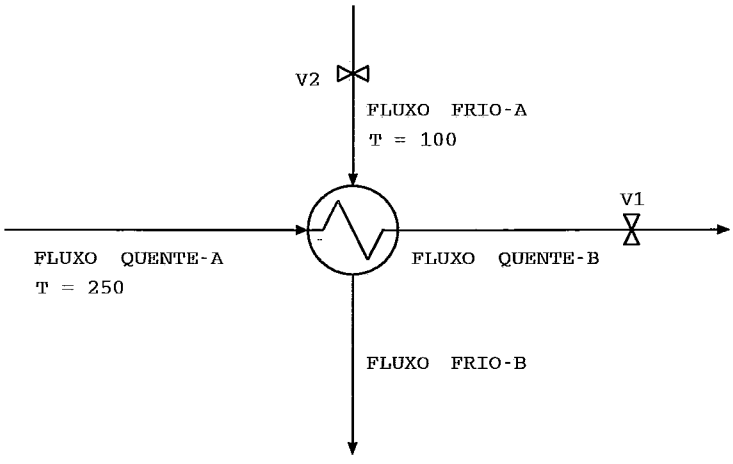

Figura 3.6: Permutador de Calor

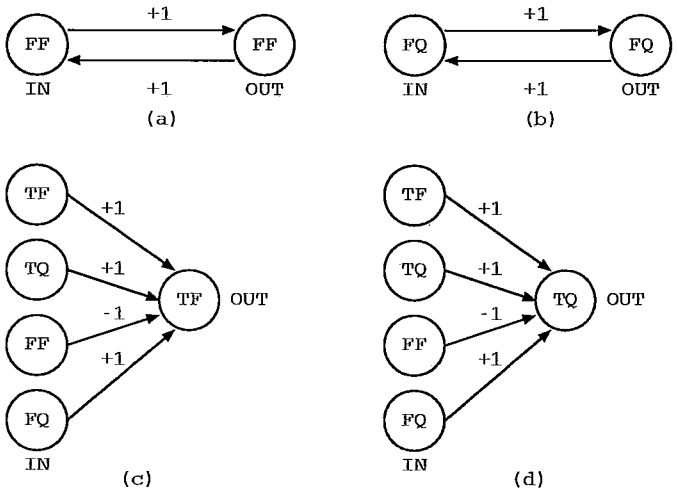

Figura 3.7: Dígrafos para o permutador de calor

$$GANHO = \frac{\Delta(\text{variável de saída})}{\Delta(\text{ variável de entrada})}$$

* Interpretação de GANHO para figura 3.7(c).
  - Se TF_{IN} aumenta/diminui TF_{OUT} aumenta/diminui e $GANHO = +1$.
  - Se TQ_{IN} aumenta/diminui TF_{OUT} aumenta/diminui e $GANHO = +1$.
  - Se FF_{IN} aumenta/diminui TF_{OUT} diminui/aumenta e $GANHO = -1$.
  - Se FQ_{IN} aumenta/diminui TF_{OUT} aumenta/diminui e $GANHO = +1$.
* **Observação:** Engloba todas as informações contidas nos vetores de diferenças e na rede IS-A.

- **Estado Inicial:** Como na figura 3.6.
- **Estado Final:** Aquele em que a meta "Reduzir temperatura de QUENTE-B" é atingida.
- **Métodos de Busca: Estratégia meio-fins[7]**
  - Como no modelo anterior, compara-se os estados inicial e final para estabelecer as variáveis a serem alteradas. Após isto tem-se uma meta.
  - Assim, suponha a meta "Reduzir a temperatura de QUENTE-B".
  - Toma-se um dígrafo em que a variável de saída seja a temperatura ou a taxa de fluxo de QUENTE-B. As duas alternativas são:
    (a) Da figura 3.7(d), a terceira relação diz que o aumento da taxa de fluxo de FRIO-A, implica na redução da temperatura de QUENTE-B, dado que $GANHO = -1$.

(b) Da figura 3.7(b): Reduzir a taxa de fluxo de QUENTE-B implica em reduzir a taxa de fluxo de QUENTE-A, e da figura 3.7(d) a quarta relação diz que a redução da taxa de fluxo de QUENTE-A implica em reduzir a temperatura de QUENTE-B, pois nos dois casos $GANHO = +1$.

- **Observação:** Os dígrafos são tomados de maneira que se tenha variáveis manipuláveis, neste caso apenas taxas de fluxo, por isso os demais dígrafos da figura 3.7(d) não foram considerados.
- Estas relações são representadas graficamente como na figura 3.8.

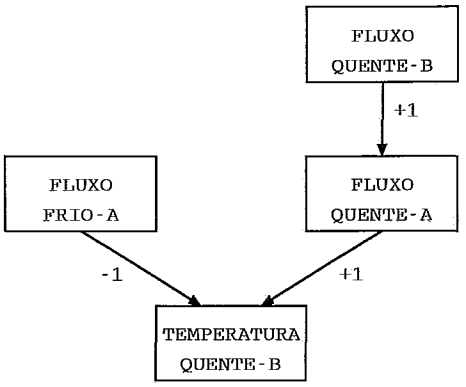

Figura 3.8: Rede de causas e efeitos para o exemplo

- Os operadores identificados para realizar esta meta seriam: "reduzir a taxa de fluxo de QUENTE-B no permutador de calor" ou "aumentar a taxa de fluxo de FRIO-A no permutador de calor".

Em relação ao primeiro modelo, a decomposição da planta em subsistemas permite alguma redução na complexidade de certas plantas, mas exige que os subsistemas sejam bem determinados devido a possíveis problemas com as interações. Os autores desta técnica ar-

gumentaram ainda que a vantagem em identificar estados estáveis é que eles podem oferecer pontos de parada para verificação e manutenção. Além disso, permitem operações em tempo real.

Através destes exemplos pode-se observar que as técnicas mais significativas são aquelas que têm uma estrutura de representação capaz de englobar todos os dados (e suas relações) que estão envolvidos no problema de *síntese de procedimentos operacionais*. Além disso, a eficiência das estratégias de busca é dependente destas representações, já que a qualidade de conhecimento e a facilidade em recuperá-lo são determinantes na construção da sequência de operações adequada.

Entretanto, embora algumas das ações consideradas tivessem a característica de contínua, estes modelos são essencialmente discretos, como já argumentado no capítulo anterior, técnicas de Planejamento são discretas, dado principalmente que as representações de conhecimento não levam em consideração o aspecto contínuo do mundo real. Mas, uma tentativa feita neste sentido pode ser vista no último modelo em que expressões numéricas são (re)introduzidas para avaliar a dinâmica contínua dos equipamentos (no caso um permutador de calor).

Assim, no próximo capítulo será apresentada uma forma de representação que oferece a possibilidade de representar de forma integrada a planta, o processo e as ações que podem ser realizadas sobre o processo. Uma outra característica é a capacidade de representar os aspectos discretos e contínuos de problemas como este.

Capítulo 4

Representação por Redes de Petri

Como discutido nos capítulos anteriores, o problema do "start-up" de plantas químicas envolve a representação de conhecimentos quantitativos e qualitativos. E por isso, na maioria dos exemplos vistos no capítulo 3, diversas formas de representação foram utilizadas na tentativa de expressar todo o conhecimento, em alguns casos, utilizou-se até mesmo uma forma de representação para cada objeto a ser representado. Mas, uma consequência desta diversidade de formas de representação é a dificuldade de recuperação e integração da informação, desde que os diferentes objetos estão intrinsecamente relacionados. E assim, também a complexidade do método de busca pela solução é aumentada.

Uma única representação que seja capaz de expressar a diversidade de conhecimento não é evidente. Contudo, a ferramenta redes de Petri possibilita a representação e integração de alguns objetos deste problema, como por exemplo o conhecimento relativo ao processo, à planta química e às ações relacionadas ao "start-up". Assim, reduz-se, em algum grau, a complexidade da representação e além disso, permite expressar aspectos simbólicos e numéricos do problema.

Este formalismo foi introduzido na década de 60 pelo matemático Carl Adam Petri, que

teve como objetivo principal desenvolver uma ferramenta capaz de descrever as relações entre condições e eventos David[34]. Mas, este não é apenas uma forma representacional esquemática para descrever regras. De fato, como argumenta Murata[31], como ferramenta gráfica, as redes de Petri podem ser usadas como um auxílio de comunicação visual similares a fluxogramas, diagramas de blocos ou redes e, como ferramenta matemática, podem estabelecer equações de estados ou algébricas e outros modelos matemáticos que governam o comportamento do sistema.

Uma característica importante deste formalismo é que a evolução ou transformação de um conjunto de condições em outro conjunto através dos eventos é determinada por uma equação de estado. Isto é, dadas as condições atuais, que definem o estado atual do sistema, e os eventos obtém-se a partir de uma equação as novas condições, ou seja, o novo estado. A relevância deste fato está em se ter uma representação formalmente verificável e comprovada. Sendo possível inclusive, determinar quais estados podem ser atingidos a partir de outros, sem que seja necessária a ocorrência dos eventos propriamente dita.

Embora se tenha outras formas de representação baseada em estados, quando se tratando de condições e eventos que podem conter aspectos simbólicos, como é o caso do "start-up", que como visto anteriormente tem tanto aspectos quantitativos quanto qualitativos a serem representados, estas formas de representação não se adaptam. De fato, tais ferramentas foram desenvolvidas visando principalmente o aspecto numérico de problemas como o "start-up". Por outro lado, as ferramentas de Inteligência Artificial são mais indicadas para representação de conhecimento simbólico, como por exemplo, a representação através

de regras de produção. Entretanto, tal ferramenta não permite avaliar "a priori" a evolução de um sistema assim representado. Isto é possível apenas através da aplicação das regras. Daí, redes de Petri se mostram capazes de relacionar os dois aspectos disjuntos do problema do "start-up".

Além disso, vale notar que esta representação tem estreita relação com Planejamento. Isto é, a capacidade de redes de Petri realizar transformações de estado através de eventos (ações) vem de encontro aos objetivos de Planejamento, a construção de planos de ação, ou seja, obter a sequência de ações capaz de conduzir o sistema de um estado inicial para o estado final desejado. Assim, todas estas características viabilizam o uso de redes de Petri para representação de sistemas dinâmicos como pode ser visto em David et al.[33] exemplos para este fato. Mas, mais do que isto, pretende-se mostrar a sua adequação em se tentar resolver um problema de Planejamento.

Entretanto, a apresentação de redes de Petri que se segue aborda apenas os seus principais aspectos básicos e que são de interesse para o trabalho desenvolvido. Sendo assim, é uma abordagem direcionada, mesmo porque este é assunto vasto, inviabilizando uma descrição completa de todas as propriedades e tipos de redes de Petri. Em Murata[31] e David et al.[33] tem-se uma extensa apresentação que dá uma visão abrangente e bem detalhada de redes de Petri.

4.1 Redes de Petri

De acordo com o mencionado anteriormente, esta ferramenta leva em consideração um conjunto de condições, um conjunto de eventos e uma equação que determina a evolução do sistema. Mas, além destes elementos, também outros são necessários. Assim, em definição 1 tem-se a formalização mais usual de redes de Petri, aquela que utiliza matrizes (Murata[31], David et al.[33], David[34]), entretanto, outras estruturas, como relações entre conjuntos (Maciel et al.[35]), podem ser utilizadas.

Definição 1: Uma rede de Petri é uma quíntupla, $RP = (P, T, Pre, Pos, M_0)$, onde

$P = \{p_0, p_2, \ldots, p_n\}$ é um conjunto não vazio e finito de *places*,

$T = \{t_0, t_2, \ldots, t_m\}$ é um conjunto não vazio e finito de transições,

com $P \cap T = \emptyset$ e $P \cup T \neq \emptyset$,

$Pre_{n \times m}$ é a matriz dos arcos de entrada $\forall t_j \in T$,

$Pos_{n \times m}$ é a matriz dos arcos de saída $\forall t_j \in T$,

$M_0 : P \rightarrow \{0, 1, 2, 3, \ldots\}$ é a marcagem(estado) inicial.

Assim, estabelecendo o paralelo, as condições são representadas pelos *places* e os eventos pelas transições. A presença de uma condição é indicada pela associação de um valor inteiro positivo, denominado *token*, ao *place* correspondente àquela condição, se este valor for 0 (nulo) a condição não está presente. Assim, um estado M_i na rede é um vetor que contém o número de *tokens* em cada *place* p_k. Daí M_0 é o vetor que representa o estado inicial da rede.

Exemplo 1:

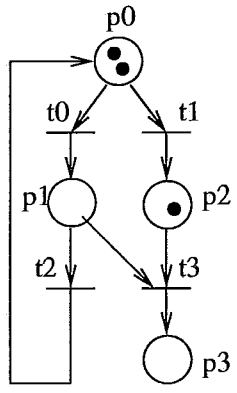

Figura 4.1: Rede de Petri Marcada

Para o exemplo 1 da figura 4.1, tem-se que:

$P = \{p_0, p_1, p_2, p_3\}$, $T = \{t_0, t_1, t_2, t_3\}$

$$Pre = \begin{array}{c|cccc} & t_0 & t_1 & t_2 & t_3 \\ p_0 & 1 & 1 & 0 & 0 \\ p_1 & 0 & 0 & 1 & 1 \\ p_2 & 0 & 0 & 0 & 1 \\ p_3 & 0 & 0 & 0 & 0 \end{array} \quad \text{e} \quad Pos = \begin{array}{c|cccc} & t_0 & t_1 & t_2 & t_3 \\ p_0 & 0 & 0 & 1 & 0 \\ p_1 & 1 & 0 & 0 & 0 \\ p_2 & 0 & 1 & 0 & 0 \\ p_3 & 0 & 0 & 0 & 1 \end{array}$$

As matrizes Pre e Pos indicam como as condições e eventos estão relacionados entre si. A primeira, Pre, identifica as condições (Pré-condições) necessárias para que os eventos ocorram e a segunda, Pos, as condições (Pós-condições) que resultarão da ocorrência do evento. Os *places* que compõem as pré-condições para uma transição são ditos *places* de entrada. E os *places* que compõem as pós-condições são ditos *places* de saída. Assim, a matriz Pre indica os *places* de entrada para cada transição, ou ainda, os arcos entre um *place* e uma transição, e a matriz Pos, os *places* de saída, ou seja, os arcos entre uma transição e um *place*. Os valores nas posições destas matrizes representam os pesos dos arcos, $w(p,t)$, para matriz Pre e os pesos $w(t,p)$ para a matriz Pos, onde o valor 0 indica a inexistência

de arco. Para o exemplo acima todos os pesos dos arcos assumem valor 1.

O estado inicial M_0 é o vetor (2 0 1 0), que representa o número de *tokens* em cada *place*, dado que $M_0(p_0) = 2$, $M_0(p_1) = 0$, $M_0(p_2) = 1$ e $M_0(p_3) = 0$. A rede de Petri do exemplo 1 é dita **marcada** devido à presença de um estado inicial em sua apresentação. Observe que entre dois *places* (duas transições) existe pelo menos uma transição (um *place*), e assim redes de Petri são também identificadas com grafos bipartites, dado que $P \cap T = \emptyset$.

As mudanças de estados que são realizadas pelas transições ocorrem de acordo com alguns critérios. O primeiro consiste em verificar se uma transição é capacitada.

Critério 1:

$$t_j \text{ é } \textbf{capacitada} \text{ se } M_i(p_k) \geq Pre(p_k, t_j) \quad \forall\, p_k \in P$$

No caso em que a transição t_j é capacitada, diz-se que t_j pode ser disparada.

Critério 2:

O **disparo** de t_j consiste em retirar $w(p, t_j)$ *tokens* de seus *places* p de entrada e adicionar $w(t_j, p)$ *tokens* a seus *places* p de saída.

O critério 2 pode ser reescrito na expressão matemática abaixo, que expressa o novo estado gerado a partir do disparo de uma transição.

$$M_i(p_k) = M_{i-1}(p_k) + W.S \quad \forall\, p_k \in P,$$

Esta é denominada a **regra de funcionamento**, ou ainda, a equação de estado mencionada anteriormente, de uma Rede de Petri, onde $W_{n \times m}$ é a matriz dada por $Pos - Pre$ e $S_{m \times 1}$ é o vetor das transições, que indica o número de vezes que cada transição $t_j \in T$ foi disparada.

Ainda de acordo com o exemplo 1, observe que as transições t_0 e t_1 são capacitadas, pois $M_0(p_k) \geq Pre(p_k, t_0)$ e $M_0(p_k) \geq Pre(p_k, t_1)$, $\forall\ p_k \in P$. Suponha então que t_0 tenha sido disparada. Assim,

$$S = \begin{matrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{matrix} \left| \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix} \right|$$

pois apenas a transição t_0 foi disparada uma única vez. E sendo,

$$W = \left| \begin{matrix} -1 & -1 & 1 & 0 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{matrix} \right|$$

o novo estado M_1 é dado por

$$M_1 = \left| \begin{matrix} 2 \\ 0 \\ 1 \\ 0 \end{matrix} \right| + \left| \begin{matrix} -1 & -1 & 1 & 0 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{matrix} \right| \times \left| \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix} \right|$$

isto é,

$$M_1 = \begin{matrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{matrix} \left| \begin{matrix} 1 \\ 1 \\ 1 \\ 0 \end{matrix} \right|$$

onde $M_1(p_0) = M_1(p_1) = M_1(p_2) = 1$ e $M_1(p_3) = 0$. Analisando este estado, tem-se que o resultado é a retirada de um *token* de p_0 e a adição deste *token* a p_1.

Considere então que M_1 é o atual estado da rede, neste caso, as transições capacitadas são t_0, t_1, t_2 e t_3. Daí, tem-se por exemplo que $M_2 = (2\ 0\ 1\ 0)$ é o estado gerado do disparo de t_2 e $M_3 = (1\ 0\ 0\ 1)$ do disparo de t_3. Observe que $M_2 = M_0$, então a sequência de transições $t_0\ t_2$ conduz ao estado inicial dado, ou seja, esta sequência provoca um ciclo que pode ser infinito. Mas, a sequência $t_0\ t_3$ conduz a um novo estado e assim não provoca ciclos.

Em M_3, as transições t_0 e t_1 são capacitadas, o disparo de t_0 gera $M_4 = (0\ 1\ 0\ 1)$ e o de t_1 gera $M_5 = (0\ 0\ 1\ 1)$. Em M_4, apenas t_2 é capacitada, gerando $M_6 = M_3$. t_3 só seria capacitada em M_4, se $M_4(p_1) > 0$ e $M_4(p_2) > 0$. Como $M_4(p_2) = 0$, t_3 não é capacitada. No estado M_5 nenhuma transição é capacitada.

Uma sequência que pode atingir M_4 a partir de M_0, por exemplo, é dada por $t_0\ t_3\ t_0$, ou seja, a sequência formada por dois disparos de t_0 e um de t_3. Assim, $S' = (2\ 0\ 0\ 1)$ é o vetor que expressa este fato. Daí,

$$M_4 = M_0 + W \times S'$$

Observe que outras sequências podem realizar este mesmo resultado, como por exemplo, as sequências $t_0\ t_2\ t_0\ t_3\ t_0$, $t_0\ t_2\ t_0\ t_2\ t_0 \ldots t_3\ t_0$ também conduzem a M_4.

Suponha agora o disparo de t_1 em M_0, o resultado é $M_7 = (1\ 0\ 2\ 0)$ e as transições capacitadas neste estado são t_0 e t_1. Do disparo de t_0 em M_7 obtém-se $M_8 = (0\ 1\ 2\ 0)$, deste podem ser gerados M_7 do disparo de t_2 e M_5 do disparo de t_3. Observe que entre outras sequências possíveis, as sequências $S_1 = t_0\ t_3\ t_1$ e $s_2 = t_1\ t_0\ t_1$ conduzem a M_5 a partir de M_0. O disparo de t_1 em M_7 gera o estado $M_9 = (0\ 0\ 3\ 0)$, no qual nenhuma transição é capacitada.

Algumas observações podem ser feitas a partir desta execução da rede. Por exemplo, diferentes sequências, s_1, s_2 e $s_3 = t_0\ t_2\ t_0\ t_3\ t_1$, que contêm ciclos ou não podem conduzir ao mesmo estado final, M_5. Sendo assim, pode-se identificar a menor ou a maior sequência que leva a um determinado estado. Entretanto, devido ao não determinismo na escolha das transições capacitadas, qualquer sequência que conduza ao estado final desejado poderá ser

gerada.

Note que o vetor S apenas expressa quantas vezes uma transição foi disparada numa dada sequência, a ordem dos disparos não pode ser identificada apenas a partir de S. Entretanto, dados um estado inicial, M_0, um estado final desejado, M_d, e um vetor S é possível verificar se M_d é atingível de M_0[1].

A transição t_3 consome *tokens*, pois retira 2 *tokens* de seus *places* de entrada, p_1 e p_2, e apenas 1 é colocado no seu *place* de saída, p_3. Mas, existe também a possibilidade de uma rede de Petri produzir *tokens*. Suponha, por exemplo, o peso $w(t_0, p_1) = 2$, isto é, o arco entre t_0 e p_1 tem valor 2. Neste caso, o disparo de t_0 retiraria apenas um *token* de p_0 e adicionaria 2 *tokens* a p_1. Assim, a cada disparo de t_0 um *token* seria produzido na rede.

Um outro ponto a observar é que existem sequências não disparáveis. A partir deste M_0 só são disparáveis, por exemplo, as sequências que começam as transições t_0 e t_1. Uma sequência que comece, por exemplo, com qualquer outra transição diferente destas duas, t_2 e t_3, não será disparável. De fato, existem várias sequências não disparáveis dependendo do estado inicial e dos estados gerados, pois para cada estado existem as transições que são e as que não são capacitadas. Assim, pode-se dizer que a cardinalidade do conjunto formado pelas sequências disparáveis e pelas sequências não disparáveis é muito grande, podendo ser infinita.

Nesta rede da figura 4.1, tem-se um exemplo de uma situação geralmente muito utilizada, a decisão. Observe que as transições t_0 e t_1 têm o mesmo place de entrada, p_0. Assim, quando

[1]Será visto com mais detalhes na próxima seção.

$M_i(p_0) > 0$ tanto t_0 quanto t_1 são capacitadas e podem ser disparadas. Esta situação é também conhecida como **conflito estrutural** pois está diretamente relacionada à topologia da rede.

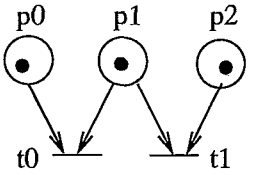

Figura 4.2: Um exemplo de conflito efetivo

Uma outra situação conflitante é o **conflito efetivo**, ou ainda, **confusão**, que pode ser vista na figura 4.2. Neste caso, se t_0 é disparada t_1 não poderá ser disparada, desde que não haverá mais *tokens* no place p_1, e vice-versa. A diferença entre as duas situações é que no primeiro caso, o conflito depende apenas da estrutura topológica da rede; no segundo, não só a estrutura da rede mas também o número de *tokens* é determinante para a existência do conflito.

Além destas duas situações acima, outras como sequenciamento, distribuição, junção, atribuição, paralelismo, mútua-exclusão e sincronismo podem ser descrita numa rede de Petri [35]. Na verdade, as redes são especificadas a partir de situações como estas. Por exemplo, nesta mesma rede da figura 4.1, t_0 e t_2 indicam sequenciamento, em t_3 tem-se uma junção. Assim, a especificação de uma rede de Petri para respresentação de um sistema deve contar com uma análise do problema e dos objetivos pretendidos para se evitar resultados indesejáveis. Pois, estes podem ocorrer devido principalmente à escolha não determinística entre as transições capacitadas num dado estado; e devido à uma situação erroneamente

representada. Em relação ao não-determinismo, condições externas podem ser acrescentadas na tentativa de evitá-lo, que seriam neste caso transições e/ou *places* extras que forçariam a ocorrência de uma determinada situação em detrimento de outras.

Um outro aspecto importante de redes de Petri é a possibilidade de identificá-las com máquinas de estado. De fato, as transições são regras que transformam os estados, assim associada à rede de Petri existe uma gramática expressa através das transições. A gramática associada à rede do exemplo 1 é dada abaixo.

$$t_0 \ : \ p_0 \rightarrow p_1$$

$$t_1 \ : \ p_0 \rightarrow p_2$$

$$t_2 \ : \ p_1 \rightarrow p_0$$

$$t_3 \ : \ p_1 p_2 \rightarrow p_3$$

Generalizando, estas regras têm o formato abaixo.

$$t_j \ : \ p_0^{x_0} \ldots p_i^{x_i} \rightarrow p_k^{x_k} \ldots p_n^{x_n}$$

onde, $p_0 \ldots p_i$ são os *places* de entrada para t_j, $x_0 \ \ldots \ x_i$ são os pesos $w(p_0, t_j) \ \ldots w(p_i, t_j)$; $p_k \ldots p_n$ são os places de saída para t_j, $x_k \ \ldots x_n$ são os pesos $w(t_j, p_k) \ \ldots w(t_j, p_n)$. Como visto, todos os arcos do exemplo acima têm pesos iguais à 1, por isso não estão representados nas regras associadas à rede.

Pode-se dizer que algumas das vantagens de redes de Petri seriam a sua formalização, simples e direta, e uma ampla capacidade representacional, pois pode ser aplicada a sistemas com características distintas. Entretanto, dois pontos críticos que podem ser citados são o não-determinismo e a possibilidade de diferentes sequências conduzirem ao mesmo estado.

4.1.1 Propriedades e Aplicações

Dois tipos de propriedades podem ser observados neste modelo básico de Redes de Petri. Um primeiro tipo está relacionado ao comportamento da rede e é dependente do estado inicial, o segundo tipo[2] relaciona-se ao aspecto estrutural da rede, ou seja, considera-se a topologia da rede, sendo independente do estado inicial.

Uma importante propriedade deste primeiro tipo é a **alcançabilidade**, que descreve os estados que podem ser alcançados a partir do estado inicial. Considerando o exemplo 1, tem-se que a partir do estado inicial M_0 dado poder-se-ía atingir dois outros, M_1, o estado resultante do disparo da transição t_0 e $M_2 = (1\ 0\ 2\ 0)$ obtido do disparo de t_1. Assim, através de disparos sucessivos das transições é possível construir um grafo de alcançabilidade, que descreve todas as possíveis sequências de transições que permitem atingir todos os possíveis estados da rede. Na figura 4.3 tem-se parte do grafo de alcançabilidade para este exemplo.

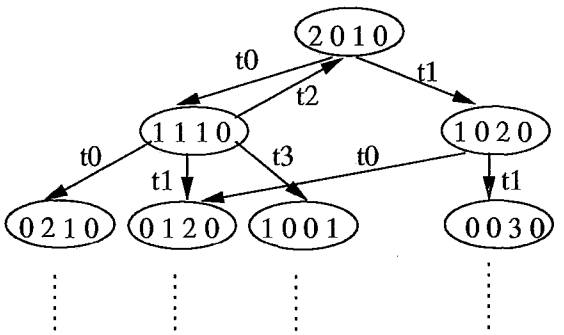

Figura 4.3: Grafo de alcançabilidade para o exemplo 1

Note que a construção de um tal grafo, ainda que se elimine os ciclos, obtendo neste caso a **árvore de alcançabilidade** (ou **de cobertura**), tem complexidade exponencial

[2]Para definições e exemplos, veja Murata et al.[31] e Maciel et al.[35].

em tempo e espaço para os casos mais gerais de redes de Petri. Contudo, tal propriedade pode ser verificada através de alguns critérios. A regra de funcionamento, por exemplo, é considerada uma condição necessária para verificar a alcançabilidade, isto é, um estado M_d é alcançável a partir de M_0, se existe um vetor S tal que $M_d = M_0 + W.S$. Para o caso em que a rede é acíclica tal condição é necessária e suficiente e assim, a verificação é imediata.

Para redes cíclicas e ordinárias é necessário acrescentar a condição de existência de *tokens* nos ciclos. De fato, se para atingir tal estado é necessário uma transição pertencente a algum ciclo, deve haver *tokens* neste ciclo, ou ainda a possibilidade de incluí-los no ciclo através de uma outra sequência de transições. Considere o exemplo 1 com estado inicial $M_0 = (0\ 0\ 1\ 1)$, o estado $M_d = (1\ 0\ 1\ 1)$ só seria atingido se existisse *tokens* no ciclo formado por p_0, t_0, p_1, t_2 e p_0, ou se existisse a possibilidade de transferir *tokens* da parte restante da rede para este ciclo. Como neste caso, nenhuma destas alternativas se verificam, não existe um vetor S que satisfaça a regra de funcionamento e daí M_d não é alcançável a partir de M_0 assim definido.

Algumas das propriedades do primeiro tipo, podem classificar as redes de Petri em:

1. **Limitadas**: Para todo estado alcançável a partir do estado inicial dado, o número de tokens em cada *place* é limitado. Observe que de acordo com os pesos associados aos arcos pode-se ter uma expansão ou redução do número de tokens.

2. **Ativas**: Para todo estado alcançável a partir do estado inicial, existem transições capacitadas, ou seja, não há "dead-lock".

3. **Puras**: Se não existe auto-ciclos entre uma transição e um *place*, isto é, se o *place* não é simultâneamente um *place* de entrada e de saída para a transição.

4. **Ordinárias**: Se todos os arcos têm pesos iguais à 1.

Além da interpretação condições/eventos para os *places* e transições, respectivamente, outras interpretações possíveis são indicadas na tabela 4.1.

Places de Entrada	Transição	Places de Saída
Pré-condições	Evento/Ação	Pós-condição
Dados de Entrada	Passo de Computação	Dados de Saída
Sinais de Entrada	Processamento de Sinal	Sinais de Saída
Condições	Cláusula em Lógica	Conclusões

Tabela 4.1: Interpretações para uma Rede de Petri

Conforme a tabela 4.1 tem-se por exemplo, que redes de Petri podem ser utilizadas para representar um algoritmo, onde os *places* de entrada representariam os dados de entrada, as transições representariam passos de computação e os *places* de saída, os dados de saída. Assim, de acordo com os exemplos dados nesta tabela pode-se enumerar um grande número de aplicações em diferentes áreas, tais como Computação de Fluxo de Dados, Comunicação de Protocolos, Controle de Sincronização, Sistemas Produtor/Consumidor, Linguagens Formais, Sistemas Multiprocessadores entre outras.

Mas, segundo Bail et al.[32], redes de Petri são basicamente um modelo discreto, isto é, o espaço de estados é discreto, permitindo apenas a modelização de sistemas de eventos discretos. Assim, deste modelo básico, que é denominado por Bail et al.[32], redes de Petri discretas, desenvolveu-se várias versões que tentam melhor se adaptar ao problema em questão, como exemplos pode-se citar as redes de Petri estocásticas, temporizadas (levam em consideração o fator tempo), contínuas (modelam eventos contínuos), híbridas (modelam

eventos discretos e contínuos) entre outras. Quando o fator tempo não está envolvido tais redes são denominadas **Autônomas**. Neste trabalho todas redes de Petri consideradas são **Autônomas**, exceto a do exemplo 2, a seguir, que é utilizada com a finalidade de apresentar redes de Petri contínuas.

4.2 Redes de Petri Contínuas

Redes de Petri contínuas[32, 33, 34] são uma extensão do modelo original, cuja introdução foi motivada por dois casos em que as redes de Petri discretas não se adapatam perfeitamente.

1. Quando eventos contínuos devem ser representados
2. Quando uma rede de Petri discreta contém um grande número de estados alcançáveis, que em geral, ocorre em redes que produzem *tokens*. Assim, redes de Petri contínuas seriam uma aproximação para este tipo de rede.

Uma forma de apresentação de redes de Petri contínuas pode ser feita através de redes de Petri discretas temporizadas. No exemplo 2, a figura 4.4 descreve uma rede de Petri discreta temporizada. Neste caso, a transição t_0 será disparada a cada intervalo de tempo $d_0 = 2$, a contar a partir do instante em que se tornar capacitada. O mesmo ocorre para t_1, a diferença está no intervalo de tempo, que é $d_1 = 3$.

Exemplo 2:

Suponha então que o estado da rede no instante $t = 0$ é como dado na figura e indicado por M_0, isto é, $M_0(p_0) = 2$ e $M_0(p_1) = 0$. Assim, no instante $t = 2$, a transição t_0 será disparada, retirando um *place* de p_0 e adicionando-o a p_1. Daí, no instante $t = 2$, t_1 torna-se

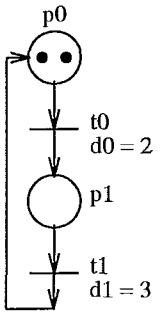

Figura 4.4: Rede de Petri discreta temporizada e marcada

capacitada, mas podendo ser disparada apenas no instante $t = 5$. Mas, t_0 pode ser disparada novamente no instante $t = 4$, antes do primeiro disparo de t_1. Assim, até $t = 4$, tem-se dois disparos sucessivos de t_0 e nenhum de t_1, sendo o estado $M_4(p_0) = 0$ e $M_4(p_1) = 2$. No instante $t = 5$, t_1 dispara, colocando um *token* em p_0, daí t_0 torna-se capacitada neste instante, podendo disparar em $t = 7$. O próximo disparo de t_1 será em $t = 8$. Desta maneira prossegue a evolução desta rede, como descrito nos gráficos da figura 4.7(a), até o instante $t = 16$.

Observe que o fator tempo já é uma tentativa de introduzir o aspecto de continuidade a redes de Petri discretas, pois entre um disparo e outro, tem-se a idéia de que algo continua acontecendo.

Uma rede de Petri contínua pode ser especificada de maneira a representar esta mesma situação, conforme figura 4.6 do exemplo 3. Observe que a notação é ligeiramente modificada para graficamente se distinguir redes de Petri contínuas de redes de Petri discretas. Esta notação de símbolos é dada na figura 4.5.

Figura 4.5: Representação gráfica na Rede de Petri Híbrida

Portanto no exemplo 3 tem-se a rede de Petri contínua, especificada a partir da rede de Petri discreta temporizada anterior. As transições contínuas t_0 e t_1 significam o número de *tokens* que são transferidos entre os *places* em função do intervalo de tempo para os disparos de t_0 e t_1 na rede discreta. Então as transições na rede contínua representam velocidades máximas de transferências de *tokens*, sendo $t_0 = V_0 = \frac{1}{2}.dt$, pois um *token* é transferido a cada 2 intervalos de tempo; e $t_1 = V_1 = \frac{1}{3}.dt$, já que se transfere um *token* a cada 3 intervalos de tempo.

O disparo de uma transição contínua, t_0 por exemplo, ocorre quando $M_t(p_0) > 0$, ou seja, o número de *tokens* no instante t deve ser maior que 0 (zero). Suponha que $t = 0$, t_0 é disparada com velocidade V_0, cujo significado é a retirada de $V_0.dt = 0.5.dt$ *tokens* de p_0 e a sua adição a p_1 entre os instantes t e $t + dt$. Assim, em $M_{t+dt}(p_1) > 0$ podendo também ocorrer o disparo de t_1. Note que $V_1 < V_0$, e isto é necessário dado que t_0 ocorre antes de t_1.

Exemplo 3:

Resumindo, as quantidades de *tokens* nos *places* p_0 e p_1 no instante de tempo t é dado por:

$$M_t(p_0) = M_0(p_0) - V_1.t + V_2.t = 2 - \frac{t}{6}$$

$$M_t(p_1) = M_0(p_1) + V_1.t - V_2.t = \frac{t}{6}$$

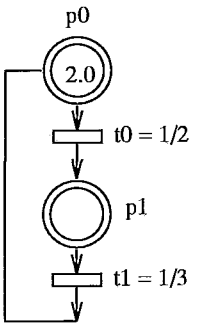

Figura 4.6: Rede de Petri Contínua Marcada

No instante $t = 12$, $M_{12}(p_0) = 0$ e $M_{12}(p_1) = 2$. Para $t > 12$, a transição t_0 não se torna capacitada, desde que $M_{12}(p_0) = 0$. Além disso, a velocidade em t_0, V_0, está limitada à velocidade de t_1, V_1, ou seja, a velocidade instantânea em t_0 é igual à velocidade instantânea (e máxima) em t_1. Daí, $V_1 = V_2$. E daí, $M_t(p_0) = 0$ e $M_t(p_1) = 2$ para $t \geq 12$. Note que mesmo quando $M_t(p_0) > 2$, a situação final será idêntica. Para Bail et al.[32], este fato é mais interessante que a rede de Petri discreta temporizada, pois nesta última, a evolução do sistema alterna entre várias fases, principalmente se o número de *tokens* for grande. E na rede de Petri contínua, a evolução alterna entre poucas fases (duas apenas para este caso), sendo portanto mais fácil a simulação do sistema através de tal rede. Os gráficos da figura 4.7(b) representam estes fatos.

De uma maneira geral, o que caracteriza as redes de Petri contínuas é a possibilidade de o número de *tokens* ser um valor pertencente ao conjunto dos reais ($\Re$). E o disparo de uma transição contínua, t_i, é a retirada de uma quantidade $x.w(p_k, t_i)$ dos seus *places* de entrada p_k e a adição de $x.w(t_i, p_j)$ a seus *places* de saída p_j, onde x é denominado a **quantidade de disparo** da transição t_i. De maneira particular, x pode ser interpretado como a transferência

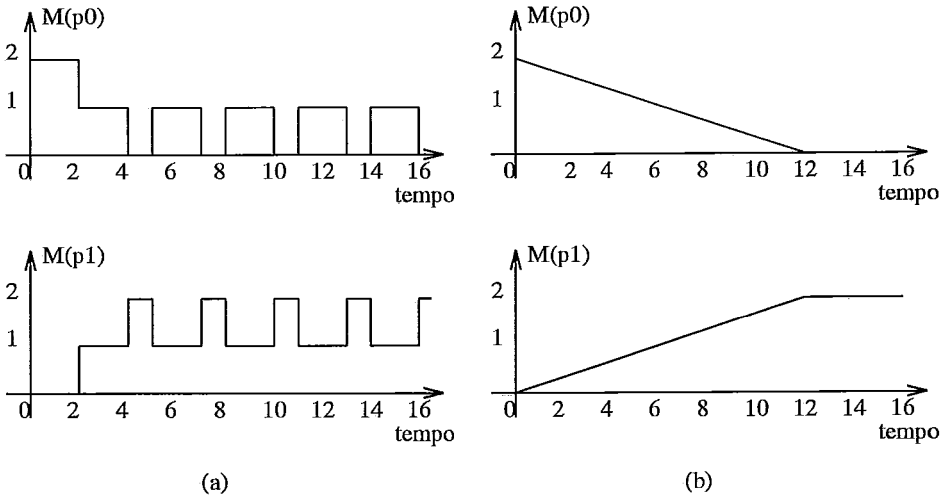

Figura 4.7: (a) Evolução da rede de Petri discreta temporizada e (b) Evolução da rede de Petri contínua

de quantidades contínuas entre dois recipientes, a produção de algum tipo de produto, ou até mesmo como a velocidade instântanea, que é o caso do exemplo acima.

Assim, a partir das definições dos dois tipos de redes de Petri, discreta e contínua, pode-se introduzir as redes de Petri híbridas.

4.3 Redes de Petri Híbridas

A principal noção envolvida em redes de Petri híbridas é a capacidade ou habilidade de englobar tanto os aspectos discretos quanto os aspectos contínuos do sistema a ser representado. Assim, o objetivo com tais redes é expandir a capacidade representacional das redes de Petri discretas, acrescentando a estas a possibilidade de também representar sistemas cuja dinâmica envolve eventos discretos e contínuos. A definição 3 é dada em Bail et al.[32], que introduziu este tipo de rede.

Definição 3: Uma rede Petri híbrida não marcada é um par $H^* =< RP, h >$ que satisfaz

as seguintes condições:

1. RP é uma rede de Petri não marcada, $RP =< P, T, Pre, Pos >$ onde,

   $P = \{p_1, p_2, \ldots, p_n\}$ é um conjunto não vazio e finito de places,

   $T = \{t_1, t_2, \ldots, t_m\}$ é um conjunto não vazio e finito de transições,

   $P \cap T = \emptyset$, isto é, P e T são disjuntos,

   $Pre : P \times T \rightarrow \{0, 1\}$ é uma aplicação de incidência de entrada,

   $Pos : P \times T \rightarrow \{0, 1\}$ é uma aplicação de incidência de saída[3]

2. $h : P \cup T \rightarrow \{D, C\}$, chamada *função híbrida*, que especifica cada *place* e cada transição, como discreto(D) ou contínuo(C).

3. As funções Pre e Pos devem satisfazer o seguinte critério: se p_i e t_j são um place e uma transição tal que $h(p_i) = D$ e $h(t_j) = C$ então $Pre(p_i, t_j) = Pos(p_i, t_j)$ deve ser verificada.

Exemplo 4:

Para o exemplo 4 acima, tem-se que:

$P = \{p_0, p_1, p_2, p_3\}$, $T = \{t_0, t_1, t_2\}$

$$Pre = \begin{array}{c|ccc} & t_0 & t_1 & t_2 \\ p_0 & 0 & 0 & 1 \\ p_1 & 0 & 0 & 0 \\ p_2 & 1 & 0 & 1 \\ p_3 & 0 & 1 & 0 \end{array} \quad \text{e} \quad Pos = \begin{array}{c|ccc} & t_0 & t_1 & t_2 \\ p_0 & 0 & 0 & 0 \\ p_1 & 0 & 0 & 1 \\ p_2 & 0 & 1 & 1 \\ p_3 & 1 & 0 & 0 \end{array}$$

[3] Pre e Pos são as matrizes como anteriormente definidas.

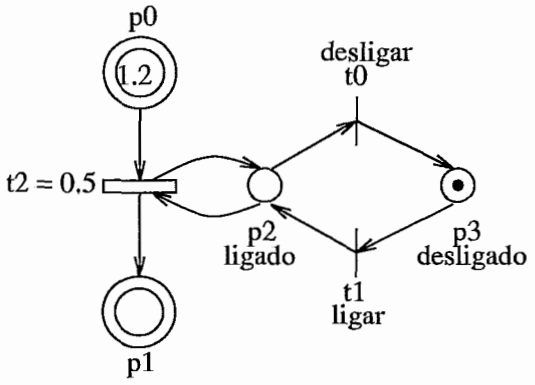

Figura 4.8: Exemplo de uma Rede de Petri Híbrida Marcada

$W = Pos - Pre$, como anteriormente. S é modificado dado à presença das transições contínuas. No componente referente à uma transição contínua deve-se colocar o número de disparos desta multiplicado pela quantidade de disparo. Sendo assim, os componentes relativos a transições discretas contêm valores inteiros e os relativos a contínuas têm valores reais.

Dado o estado $M_0 = (1.2\ 0\ 0\ 1)$ apresentado na rede da figura 4.8, a única transição capacitada é t_1. Suponha então o disparo de t_1, seguido de dois disparos sucessivos de t_2, ou seja, a sequência $t_1\ t_2\ t_2$. Assim,

$$S = \begin{array}{c|c} t_0 & 0 \\ t_1 & 1 \\ t_2 & 1.0 \end{array}$$

e o estado M_1 obtido a partir do estado M_0 é como abaixo:

$$M_1 = \begin{vmatrix} 1.2 \\ 0 \\ 0 \\ 1 \end{vmatrix} + \begin{vmatrix} 0 & 0 & -1 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ 1 & -1 & 0 \end{vmatrix} \times \begin{vmatrix} 0 \\ 1 \\ 1.0 \end{vmatrix}$$

isto é,

$$M_1 = \begin{vmatrix} 0.2 \\ 1.0 \\ 1 \\ 0 \end{vmatrix}$$

onde $M_1(p_0) = 0.2$, $M_1(p_1) = 1.0$, $M_1(p_2) = 1$ e $M_1(p_3) = 0$.

O estado M_0 diz que a máquina está desligada, não sendo possível a sua operacionalização. Então é necessário primeiro ligá-la para depois realizar a produção, transferência ou qualquer outro evento (ação) contínua relativa a t_2. Assim, o estado M_1 expressa este fato.

A transição contínua não altera o número de *tokens* nos seus *places* de entrada discretos (também seus *places* de saída). Isto está especificado no item 3 da definição. Observe que $Pre(p_2, t_2) = Pos(p_2, t_2) = 1$.

Os critérios de transição capacitada e de disparo para transições discretas é como antes, mesmo que haja *places* de entrada/saída contínuos. Entretanto, uma transição contínua, t_j é capacitada se para os *places* discretos é verificada o critério 1 anterior e se para os *places* contínuos, o número de *tokens* for maior que zero, isto é,

1. se $\forall\ p_k$, com $h(p_k) = D$ e p_k é *place* de entrada para t_j, $M_i(p_k, t_j) \geq Pre(p_k, t_j)$.

2. se $\forall\ p_k$, com $h(p_k) = C$ e p_k é *places* de entrada para t_j, $M_i(p_k, t_j) > 0$.

Na verdade, as transições discretas podem transformar *tokens* reais em inteiros e vice-versa, por isso pode haver qualquer tipo de *place* ligado a qualquer transição discreta.

Observe que as transições discretas estão relacionadas a eventos que refletem apenas uma mudança de estado, onde cada transição discreta relaciona dois estados possíveis. No exemplo acima as transições ligar e desligar podem promover os estados ligado ou desligado, respectivamente. As transições contínuas refletem eventos em que seja possível mais que dois estados, e por isso, manipulam *tokens* que tomam valores no conjunto dos reais. Sendo assim, tais eventos podem ser vistos como situações que continuam no tempo, ou ainda, que

levam um período de tempo mais longo para serem realizadas. Mas, vale ainda lembrar que o fator tempo não está sendo considerado.

Como visto, a equação fundamental ou regra de funcionamento é a mesma, tanto para redes de Petri discretas quanto para as redes híbridas, daí algumas propriedades verificadas para as redes de Petri discretas são também verificadas para as híbridas.

4.4 Modelagem com Redes de Petri Híbridas

Considere a planta química representada na figura 4.9, retirada de Fusillo et al.[11], onde foi utilizada como exemplo para verificar o modelo por eles desenvolvidos (veja capítulo 3, Modelos de Fusillo, Segundo Modelo) e que contém todos os dados necessários para uma boa formulação do problema. Um outro motivo para também tomar este exemplo é o fato de redes de permutadores serem estudadas de modo particular, dada a sua importância e a grande utilização em indústrias químicas.

Nesta planta da figura 4.9 tem-se rede de permutadores de calor, onde $HX1$, $HX2$, $HX3$ e $HX4$ representam os permutadores, $F1$, $F2$, $F3$ e $F4$ representam as fontes dos fluxos, e $S1$, $S2$, $S3$ e $S4$ representam os sumidouros dos fluxos. Alguns fluxos como $H11$, $C12$ e $C22$ são fluxos de saída do processo. As temperaturas dos fluxos de entrada também são indicadas.

Na planta da figura 4.9, o principal objetivo é alterar as temperaturas das substâncias envolvidas, ou ainda, trazer as substâncias de entrada para as temperaturas desejadas, ou até mesmo obter novas substâncias através da mudança de temperatura, como pode ser

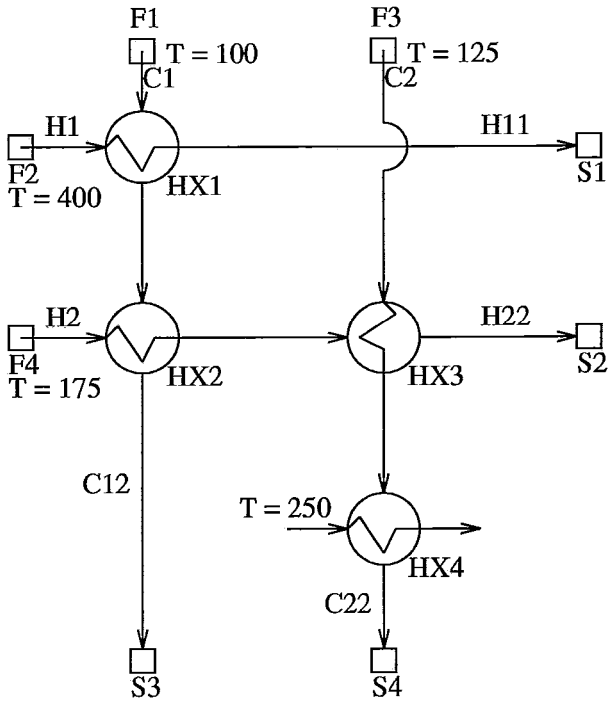

Figura 4.9: Fragmento de uma Planta Química

visto em indústrias petroquímicas. Sendo assim, uma rede de permutadores, em geral, faz parte de um processo mais amplo no qual o controle de temperaturas e de quantidades de fluxo é apenas uma etapa do processo. Mas, não diminuindo a sua importância, dado que diferentes topologias, objetivos e restrições estão envolvidos na construção de uma rede de permutadores.

Como visto em outros modelos apresentados no capítulo 3, o processo descrito numa planta química tem um caminho a ser seguido. De uma certa forma tem-se uma sequência de subprocessos sequenciais e/ou paralelos que devem ocorrer para que o processo chegue ao fim. Na planta da figura 4.9 pode-se observar que todos os quatro permutadores estão dispostos de forma sequencial, ou seja, $HX4$ será inicializado após $HX3$, este será inicializado

após $HX2$ e finalmente este será inicializado após $HX1$.

Desta maneira, quatro subprocessos sequenciais podem ser identificados correspondentes a cada permutador de calor, que compõem o processo geral de toda a planta. Contudo, vale notar que tal sequenciamento existe nos instantes de inicialização do processo na planta e que após isto todos ocorrem de forma paralela. O "start-up" do processo representado nesta rede é o "start-up" dos subprocessos 1 ($HX1$), 2 ($HX2$), 3 ($HX3$) e 4 ($HX4$), nesta ordem. Assim, a representação geral de todo o processo, figura 4.10, pode ser feita a partir da representação dos subprocessos.

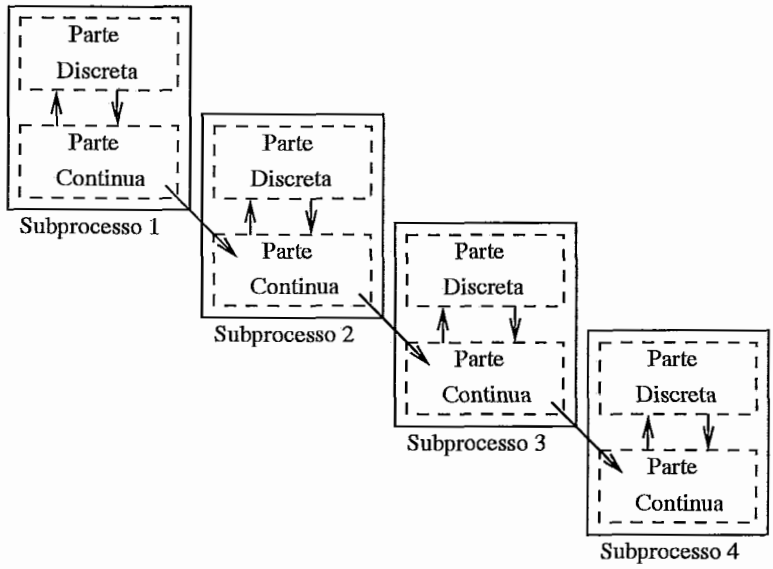

Figura 4.10: Esquema Geral da Rede de Petri Híbrida para o exemplo

A realização de cada um destes subprocessos pode envolver ações discretas, como por exemplo, verificar valores das quantidades físicas presentes em cada equipamento do subprocesso, verificar o estado físico dos equipamentos e verificar o estado operacional dos equipamentos; e ações contínuas, tais como, alterar (aumentar ou reduzir) quantidades físicas (tem-

peratura e fluxo). Assim, para o exemplo considerado, tem-se quatro subprocessos idênticos, onde a representação[4] dos subprocessos 1 e 2 é dada na figura 4.11. Observe ainda, que as ações contínuas devem, preferencialmente, ocorrer após as discretas no momento da inicialização do processo, desde que algumas verificações devem ocorrer antes da operacionalização dos equipamentos.

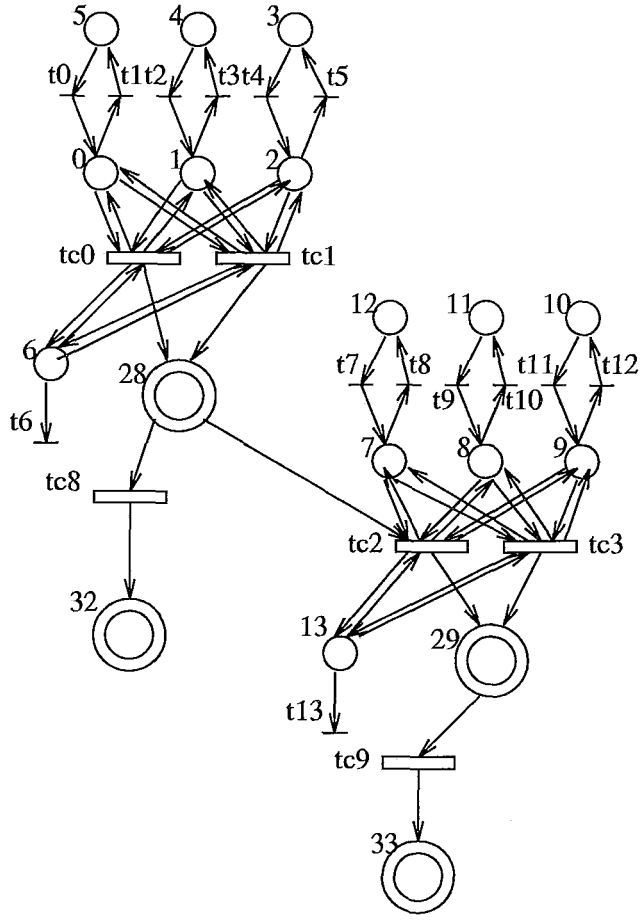

Figura 4.11: Rede de Petri híbrida para os subprocessos 1 e 2

Baseado no fato de que um subprocesso pode ser visto como uma (ou mais) ação(ões) contínua(s), considerou-se neste caso do permutador de calor, duas ações contínuas, o fluxo

[4]Para simplificação da figura, apenas dois subprocessos foram representados.

da substância quente (aquecimento) e o fluxo da substância fria (resfriamento). Assim, no subprocesso 1, as transições tc_0 e tc_1 são as transições contínuas que representam o aquecimento e o resfriamento, ou ainda a entrada do fluxo quente, $H1$, e do fluxo frio, $C1$ no permutador de calor $HX1$ representado pelo *place* contínuo 28. A transição tc_8 representa o fluxo da substância quente, H_{11}, resultante da troca de calor para o *place* contínuo 32, que seria o fluxo de H_{11} para o sumidouro S_1 na figura 4.9.

A transição discreta t_6 e o *place* discreto 6 estão caracterizando o fim do subprocesso 1, ou seja, o fato de t_6 estar capacitada significa que o subprocesso 1 está em evolução, em caso contrário algum problema provocou a sua parada. Assim, esta transição oferece um ponto de verificação da evolução do subprocesso. Observe que este *place* pode ser visto como um ponto de conexão entre o processo físico ocorrendo na planta e a representação deste na rede de Petri, isto seria então um ponto para manter um sistema de supervisão de processos.

Como existem mais dois outros subprocessos, existem dois outros *places* discretos, 20 e 27; e duas outras transições discretas, t_{20} e t_{27} que determinam os finais dos subprocessos 3 e 4, respectivamente. E todas estas quatro transições se influenciam mutuamente, pois desde que os quatro subprocessos são sequenciais, a parada de um deve influenciar a parada de outro.

As transições discretas t_0, t_1, t_2, t_3, t_4 e t_5 identificam estados operacionais. t_0 por exemplo pode representar a ação de abrir válvulas das fontes F_1 e F_2, e t_1 fechar; t_2 abrir válvula do sumidouro S_1 e t_3 fechar; t_4 ligar permutador $HX1$ e t_5 desligar. Observe que se não houver *tokens* em pelo menos um dos *places* 0, 1 e 2, as transições contínuas, tc_0 e

tc_1, não podem disparar, pois estas são capacitadas quando o número de *tokens* nos seus *places* de entrada discretos é naior que zero (0). A existência de *tokens* nestes *places* significa que é possível operacionalizar o permutador $HX1$, isto é as condições verificadas para seu funcionamento estão garantidas.

Existem três ciclos de transiçõcs discretas em cada subprocesso. Cada ciclo representa uma situação em que as duas transições discretas presentes se opõem. Mas, as três situações são independentes entre si, ou seja, a ordem em que as transições dos três ciclos aparecem não é relevante. Ou ainda, as sequências $t_0\ t_2\ t_4$ ou $t_4\ t_0\ t_2$ ou qualquer outra variação destas três transições, por exemplo, não é relevante.

De fato, a ordem que se deseja preservar já está expressa na especificação da rede, que é a ordem dos subprocessos, supõe-se assim, que dentro de cada subprocesso, tanto as transições discretas quanto as transições contínuas podem ocorrer em paralelo.

Na verdade, foi considerado na especificação desta rede que as transições representam ações tão complexas quanto se queira. Isto é, elas são mais que simples ações, são de fato consideradas procedimentos. Sendo assim, a dependência entre condições e situações está expressa nos procedimentos relativos às transições.

Por exemplo, a transição t_0, que pode significar abertura de válvula, pode depender da condição da fonte a que pertence, pois esta fonte estar vazia. Não fazendo sentido portanto, a abertura de tal válvula. Logo, esta transição pode ser um procedimento.

abrir_válvula(válvula, fonte):-

se verificar_fluxo(fonte)

De fato tais suposições estão baseadas no fato de que a representação de todas as condições e situações conduziria a uma rede complexa. E a representação de condições dependentes aumentaria ainda mais esta complexidade.

Observe que uma fonte para o permutador $HX2$ é uma saída do permutador $HX1$. Sendo assim, o subprocesso 2 está ligado ao subprocesso 1 através de uma transição contínua, tc_2, que é simultaneamente a saída do fluxo frio de $HX1$ (*place* contínuo 28) e uma entrada para o permutador $HX2$ (*place* contínuo 29). tc_3 é a transição contínua representando o fluxo quente da fonte F_3 para o permutador $HX2$.

Daí, os processos 2, 3 e 4 são idênticos ao subprocesso 1, tendo sempre uma transição contínua unindo-os e também as transições discretas descritas acima, apenas rotuladas diferentemente. Uma representação simplificada de cada um destes subprocessos é dada abaixo.

- **Subprocesso 1**:

  1. Conjunto de Transições discretas: $TD_1 = \{t_0, t_1, t_2, t_3, t_4, t_5\}$
  2. Conjunto de Transições contínuas: $TC_1 = \{tc_0, tc_1, tc_8\}$
  3. Conjunto de *Places* discretos: $PD_1 = \{0, 1, 2, 3, 4, 5\}$
  4. Conjunto de *Places* contínuos: $PC_1 = \{28, 32\}$

- **Subprocesso 2**:

  1. Conjunto de Transições discretas: $TD_2 = \{t_7, t_8, t_9, t_{10}, t_{11}, t_{12}\}$

2. Conjunto de Transições contínuas: $TC_2 = \{tc_2, tc_3, tc_9\}$

3. Conjunto de *Places* discretos: $PD_2 = \{7, 8, 9, 10, 11, 12\}$

4. Conjunto de *Places* contínuos: $PC_2 = \{29, 33\}$

- **Subprocesso 3**:

1. Conjunto de Transições discretas: $TD_3 = \{t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}\}$

2. Conjunto de Transições contínuas: $TC_3 = \{tc_4, tc_5, tc_{10}\}$

3. Conjunto de *Places* discretos: $PD_3 = \{14, 15, 16, 17, 18, 19\}$

4. Conjunto de *Places* contínuos: $PC_3 = \{30, 34\}$

- **Subprocesso 4**:

1. Conjunto de Transições discretas: $TD_4 = \{t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}\}$

2. Conjunto de Transições contínuas: $TC_4 = \{tc_6, tc_7, tc_{11}\}$

3. Conjunto de *Places* discretos: $PD_4 = \{21, 22, 23, 24, 25, 26\}$

4. Conjunto de *Places* contínuos: $PC_4 = \{31, 35\}$

Assim, o conjunto dos *places* discretos, $PD = PD_1 \cup PD_2 \cup PD_3 \cup PD_4$ e o número de *places* discretos é $NPd = 28$. O conjunto dos *places* contínuos, $PC = PC_1 \cup PC_2 \cup PC_3 \cup PC_4$ e o número de *places* contínuos é dado por $NPc = 8$, o conjunto das transições discretas, $TD = TD_1 \cup TD_2 \cup TD_3 \cup TD_4$ e o número de transições discretas é dado por $NTd = 28$ e o conjunto das transições contínuas, $TC = TC_1 \cup TC_2 \cup TC_3 \cup TC_4$ e o número transições

contínuas é $NTc = 12$. As transições contínuas tc_2, tc_4 e tc_6 fazem a ligação entre os subprocessos. A ordem em que foi dadas as transições e os *places* corresponde a ordem em que eles aparecem de forma idêntica no subprocesso 1.

Em relação ao caminho do fluxo do processo, este fica determinado durante a especificação da rede de Petri, dado que a rede pressupõe o conhecimento dos subprocessos necessários para realização do processo desejado.

Entretanto, este aspecto pode ser tanto generalizado quanto automático. Pois, no momento de construção de uma planta, sabe-se todos os processos que podem ser realizados a partir dela. E mais, sabe-se também quais são os subprocessos ou submetas ou caminho de fluxo de cada processo, logo tal rede de Petri pode ser conjuntamente construída no momento da construção da planta. A rede não é a topologia da planta, mas o fluxo dos processos em relação à topologia da planta.

No estudo feito, a construção da rede não se deu de forma automática, contudo pode-se observar que esta é uma tarefa que pode ser realizada, dado que a rede é composta de características simples e gerais: as transições discretas são operações como verificações que podem ser determinadas de acordo com cada subprocesso identificado levando em consideração os seus aspectos químicos e os equipamentos nele contido; as transições contínuas estão relacionadas com o funcionamento propriamente dito de cada equipamento presente no processo.

Se comparado ao método de decomposição da planta em subsistemas (subprocessos) que contenham estados estacionários ou estáveis, o método apresentado é menos exigente,

pois toma essencialmente um equipamento como um subprocesso, mas evita os problemas de interações entre os subprocessos, dado que identifica a sequencialidade e o paralelismo presentes no fluxo do processo e é segundo estas características que o fluxo é determinado.

Observe que as transições contínuas em cada subprocesso deveriam ocorrer o máximo possível em paralelo para evitar a violação das restrições de produção.

Uma das vantagens que se obtém a partir deste modelo de representação é a possibilidade de descrever o caminho do processo, as operações relacionadas a cada subprocesso e aos equipamentos e algumas características operacionais, simultaneamente. Isto é, tudo está representado dentro do mesmo formalismo, não sendo necessário portanto, a utilização de diferentes formas de representação do conhecimento para descrever todas as diferentes informações relativas a este tipo de problema, como pode ser visto em alguns modelos apresentados no capítulo 3.

Capítulo 5

Algoritmos Genéticos

A representação utilizada e apresentada no capítulo 4 permite a identificação de sequências de transições que conduzem o sistema de um estado inicial dado para outro estado. Então com base nesta representação, pode-se obter uma sequência de transições que permita atingir um estado final a partir de um estado inicial dado. Isto é, a partir desta representação é possível obter um plano de ação. Assim, dada a rede de Petri que representa o processo e a planta, deseja-se obter um plano, ou seja, uma sequência de transições que identifica os procedimentos operacionais para o "start-up" de plantas químicas. Note, entretanto, que as transições na sequência devem estar ordenadas para evitar situações indesejadas.

Mas, devido ao não determinismo da evolução de redes de Petri, a identificação de uma sequência capaz de realizar o estado final desejado a partir do estado inicial dado, pode ser complexa, lenta e no pior caso esta evolução pode entrar num ciclo infinito.

Além disso, uma alternativa para tal realização seria a construção da árvore de alcançabilidade, que garantiria a identificação da(s) sequência(s) ordenada(s) de transição(ões) que representariam o plano de ação desejado. Mas, a complexidade de tal tarefa é exponencial em tempo e espaço. Uma outra alternativa seria obter o vetor S de transições disparadas

a partir do estado inicial dado e do estado final a ser atingido. Entretanto, este vetor não especifica a ordenação das transições na sequência por ele representada.

Portanto, tem-se a necessidade de utilizar um método de busca próprio para grandes espaços de busca, pois como visto, o espaço formado pelas sequências, disparáveis ou não, é grande. Algoritmos Genéticos são métodos de busca, que entre outras características, tenta suprir uma certa ineficiência de métodos anteriores em relação a este aspecto do espaço de busca.

Uma importante motivação em torno deste tipo de algoritmo vem do objetivo de se tentar construir máquinas ou sistemas inteligentes, que sejam capazes de simular o comportamento humano. Pode-se dizer que a introdução de paradigmas computacionais baseados em teorias biológicas sobre tal comportamento é relativamente recente. O paradigma das Redes Neuronais Artificiais, por exemplo, foi proposto na década de 50 com objetivo de traduzir a formação do conhecimento humano. E a partir da última década tem sido aplicado a diferentes áreas do conhecimento.

Algoritmos Genéticos também podem ser citados como exemplos para este fato. Proposto na década de 70 visando ser uma alternativa para solução de problemas anteriormente resolvidos por métodos, como o Método Gradiente. As duas principais características destes problemas que os algoritmos genéticos tentaram superar foram a presença de um grande espaço de soluções e a necessidade de um grande número de cálculos. Além disso, também se propunha a resolver problemas, cujas soluções, em geral, tendiam para mínimos locais.

Várias aplicações[14] destes algoritmos em diferentes áreas do conhecimento foram reali-

zadas, incentivando o seu uso e o seu desenvolvimento. Introduziu-se assim, diversas versões diferenciadas da original, que contêm modificações derivadas das particularidades do problema sendo tratado. Pode-se dizer que estas também contribuiram para a evolução teórica e prática destes algoritmos. A seguir será apresentada a sua versão original, também dita padrão [18].

5.1 Descrição do Algoritmo

Baseado nas teorias da evolução e seleção natural de Darwin, J. H. Holland apresentou em seu livro *Adaptation in Natural and Artificial Systems (1975)* os princípios básicos de um modelo computacional para a resolução de problemas, denominado *Algoritmo Genético*. Segundo estas teorias biológicas, uma população natural de indivíduos evolui, através de gerações sucessivas, de acordo com mecanismos seletivos que permitem apenas a sobrevivência dos indivíduos melhores adaptados ao ambiente. A partir disto derivou conceitos correspondentes e similares a estes para compor tal método.

Assim, dado que um indivíduo pode ser identificado por uma cadeia de genes ou cromossoma, onde cada gen expressa uma característica do indivíduo, associou a isto uma cadeia de bits para representar o indivíduo artificial. Como cada gene está relacionado a dois alelos, valores que indicam a presença ou ausência de determinada característica, tomou os valores 0 e 1 para corresponder aos alelos biológicos. Logo, um indivíduo neste modelo artificial é uma cadeia de 0's e 1's e uma população é um conjunto destas cadeias.

Os mecanismos seletivos realizam as mudanças, que determinam a evolução de uma

população, através das gerações. Tais mudanças podem ocorrer devido às interações entre os indivíduos ou devido às influências do ambiente sobre o indivíduo. Disto derivou três mecanismos básicos, cruzamento, reprodução e mutação, que foram denominados *operadores genéticos*, para realizar a evolução do algoritmo. A aplicação destes operadores é precedida de um processo de seleção dos indivíduos melhores adaptados, que utiliza uma função para avaliação dos indivíduos denominada *função de adaptação.*

O operador cruzamento realiza trocas entre os indivíduos, de acordo com os seguintes passos:

1. seleciona-se aleatoriamente entre os indivíduos melhores adaptados dois indivíduos
   - Exemplo:
   - i_1 : 10111011
   - i_2 : 01011100
2. determina-se aleatoriamente um ponto nestes indivíduos, denominado *ponto de cruzamento*, como por exemplo entre as posições 3 e 4.
   - i_1 : 101|11011
   - i_2 : 010|11100
3. troca-se as duas subcadeias formadas a partir do ponto de cruzamento produzindo dois novos indivíduos que são adicionados à população seguinte
   - i_1 : 101|11100

- i_1 : 010|11011

A reprodução induz à proliferação de indivíduos bem adaptados. Este operador foi identificado com a adição de cópias de indivíduos aleatoriamente selecionados de acordo com a sua adaptação na população seguinte.

A mutação realiza alterações nos bits dos indivíduos, da seguinte forma:

1. seleciona-se aleatoriamente entre os indivíduos melhores adaptados um indivíduo

   - Exemplo:

   - i_1 : 10111011

2. determina-se aleatoriamente um ponto neste indivíduo, como por exemplo a posição 5.

   - i_1 : 1011|1|011

3. troca-se o valor deste bit de 0 para 1 ou de 1 para 0, dependendo do seu valor, produzindo um novo indivíduo que é adicionado à população seguinte

   - i_1 : 1011|0|100

Segundo teorias biológicas tem-se que estes operadores não ocorrem todos com a mesma frequência. Por exemplo, diz-se que dentre os três, a mutação ocorre raramente, em oposição ao cruzamento. Logo, estabeleceu-se para cada operador um valor de probabilidade. Assim, a aplicação de qualquer deles é decidida segundo uma seleção baseada nestas probabilidades, p_m para mutação, p_c para cruzamento e p_r para reprodução. Na figura 5.1 tem-se o algoritmo

```
procedure AG
begin
     t := 0;
     inicialize P(t);
     avalie estruturas em P(t);
     while  nao condicao de termino  do
     begin
          t := t + 1;
          selecione P(t) de P(t - 1);
          aplique operadores geneticos nas
                  estruturas em P(t);
          avalie estruturas em P(t);
     end
end
```

Figura 5.1: Algoritmo para um AG padrão

geral na versão original. Observe que o número de gerações em que o algoritmo evolui é a condição de parada do algoritmo.

De uma forma geral, as dificuldades que envolvem este algoritmo são as determinações da função de adaptação, do comprimento da cadeia e do tamanho da população, desde que são dependentes do problema. Em relação a estes dois últimos, tem-se que o tamanho da população deveria ser proporcional ao comprimento do indivíduo. Pois quanto maior o comprimento do indivíduo maior o espaço de soluções, sendo assim o tamanho da população deve ser expressivo em relação a este espaço. Caso assim não seja, a convergência[1] do algoritmo pode ser ou lenta ou prematura ou mesmo não ocorrer. Por outro, populações grandes podem gerar problemas computacionais. Embora, existam teorias e pesquisas que

[1]É necessário dizer, que apesar da teoria desenvolvida em torno dos esquematas para provar a convergência do algoritmo, esta na verdade não é confirmada. Sendo assim, não há garantias quanto a convergência.

apresentam intervalos para a variação do tamanho da população e o número de gerações, estes são dados empíricos.

5.2 Aplicações

A classe de problemas que mais se adaptou a esta primeira versão foi aquela que envolvia otimização de parâmetros[15]. Neste caso, tem-se um conjunto de parâmetros, para os quais deve-se determinar valores, de acordo uma função objetivo. Em geral, é um problema de minimizar ou maximizar esta função. O mapeamento deste problema para o modelo de um algoritmo genético, onde os indivíduos são representados por sequências de "bits", é imediato. Neste caso, basta representar os parâmetros do problema pelos "bits" que formam os indivíduos. A função de adaptação é a própria função objetivo do problema. Caso seja uma minimização, a solução será o indivíduo que tiver o menor valor para esta função, após a evolução do algoritmo segundo um número previamente estabelecido de gerações.

Para a aplicação dos algoritmos genéticos é necessário que se reconheça e extraia do problema as características relevantes que determinam a sua solução de forma que seja possível colocá-las na representação de cadeias de bits. Entretanto, para algumas classes de problemas este mapeamento do problema para esta forma de representação pode não ser evidente. Assim, outras formas de representação foram popostas, introduzindo novas versões para o algoritmo.

Um exemplo é a aplicação de AGs a problemas com parâmetros variando em intervalos contínuos. Grefensttete[15] propôs um algoritmo onde técnicas de discretização foram em-

pregadas para atingir a cadeia de bits. Recentemente um novo algoritmo[2] para este mesmo tipo de problema foi introduzido por XQi et all.[37] e revisado por Fogel[38], em que os bits nos indivíduos são valores reais e a função de adaptação leva em consideração uma função penalidade para a violação das restrições.

Seguindo a alternativa de discretização, o Sistema Classificador de Booker et al.[22] é uma proposta para aprendizagem de regras que evolui através de um algortimo genético. As regras de formato IF <condição> THEN <ação> foram descritas utilizando cadeias de bits, tanto para a parte IF quanto para a parte THEN. Assim, a concatenação destas duas cadeias formavam a representação de uma regra.

Sistemas como SAMUEL [16] utilizaram as regras no próprio formato IF <condição> THEN <ação> para aprendizagem de planos com multiagentes. Um outro exemplo nesta mesma linha é a geração automática de programas LISP introduzida por Fujiki[23] e De Jong[24], em que as cadeias eram *S*-expressões desta linguagem.

Devido a estas alterações na representação também os operadores foram alterados. Por exemplo, no caso do modelo de De Jong, o cruzamento deveria preservar as regras sintáticas de geração das cadeias, dado que as geração das cadeias da população inicial se dava de acordo com a gramática da linguagem LISP. Assim, os pontos de cruzamento deveriam ser selecionados em pontos mais específicos. Os demais operadores permaneceram como propostos. Entretanto, novos tipos foram introduzidos, como a inversão, que determina aleatoriamente uma subcadeia de um indivíduo e a inverte. Por exemplo, dado o indivíduo

[2]Será visto com mais detalhes no próximo capítulo.

com a subcadeia delimitada por "|", 101|110|00, o indivíduo resultante seria 101|011|00. Este é um operador interessante para problemas em que a ordem dos bits na cadeia é significativa. Nos dois últimos exemplos a ordem das regras e dos procedimentos garantem a coerência do programa.

Uma outra alteração, decorrente destas versões, é a forma de avaliação dos indivíduos. Nestes casos acima, a função de adaptação leva em consideração a simulação do indivíduo, desde que este é um programa ou um procedimento. Entretanto, ao se utilizar LISP, este simulador é imediato, desde que cada inidivíduo já é o próprio programa. O crescimento das pesquisas nesta linha determinaram o surgimento da área de Programação Genética, cujo principal objetivo é a geração automática de programas através de Algoritmos Genéticos[25, 26, 27].

Ainda sobre as alterações, pode-se citar outras, como por exemplo, a inclusão de conhecimento específico do problema nos operadores genéticos[17], o cruzamento ocorrendo em vários pontos do indivíduo e a eliminação do operador mutação, devido a sua baixa probabilidade (Fogel[38]). Todos estes acréscimos vêm permitindo a expansão do espectro de atuação de Algoritmos Genéticos.

Capítulo 6

Um Algoritmo Genético para busca em Rede de Petri Híbrida

A solução do problema é uma sequência de operações ou ações, assim os indivíduos que constituem as populações do algoritmo genético são sequências de operações. No caso da representação adotada estas operações são representadas pelas transições discretas ou contínuas da rede de Petri híbrida. Daí, a partir da representação do processo como descrito nas figuras 4.10 e 4.11 através de uma rede de Petri híbrida, um algoritmo genético é desenvolvido de maneira a obter uma sequência ótima que estabeleça o estado final desejado, ou seja, o estado em que ocorra o "start-up".

A busca pela solução foi dividida em duas etapas, a busca e otimização da parte discreta e a busca e otimização da parte contínua. Isto devido à dependência da parte contínua em relação à discreta. Além disso, existe uma diferença de objetivos entre as duas etapas. Na discreta, deseja-se encontrar a sequência de operações, mas na parte contínua, supõe-se que todas as transições contínuas sejam necessárias, desde que representam o processo propriamente dito, restando assim, apenas a determinação da quantidade de disparo de cada

uma delas.

Em um possível sistema de supervisão, a divisão da solução do problema pode facilitar, desde que pode haver problemas relativos apenas à parte discreta, ou apenas relativo à parte contínua. Assim, seria mais rápido e eficiente tratar apenas da parte envolvida.

Um método de otimização é necessário, pois a evolução da rede de Petri é não determinística e além disso, a sequência de transições resultante pode não ser a ótima segundo algum critério de otimalidade. Um critério de otimalidade, neste caso, é a obtenção da menor sequência que atinja o estado final desejado, desde que se pretende ter algum grau de eficiência.

Assim, dadas estas características do problema e da sua solução, algoritmos genéticos podem ser utilizados, pois são métodos de busca e otimização. Na parte discreta, utilizou-se a versão original e também uma nova versão é introduzida. Para a parte contínua utilizou-se um algoritmo genético para otimização com restrições (Homafair[37] e Fogel[38]). Pois nesta parte podem haver diversas restrições quanto à produção das substâncias, como por exemplo, restrições quanto à temperatura e quantidade de fluxo.

6.1 Busca e Otimização da Parte Discreta

Neste caso, considera-se apenas a parte discreta da representação, isto é, as transições discretas. Duas versões do algoritmo são apresentadas nesta etapa, onde a origem da diferença se encontra na forma de geração da população inicial. E como consequência para cada uma destas duas formas de geração, diferentes operadores genéticos foram utilizados e também

diferentes funções de adaptação. A primeira forma de geração está baseada no algoritmo genético padrão, ou seja, a geração aleatória da população inicial e os operadores cruzamento, mutação e reprodução. A segunda forma baseia-se na gramática associada à rede de Petri híbrida utilizada para a representação do problema.

6.1.1 Proposta de um Algoritmo Genético Padrão

Para a formulação do algoritmo são necessários alguns dados de entrada. Um primeiro dado é a população inicial, que é composta de sequências de transições discretas. Assim, é uma geração aleatória tanto na seleção das transições que formam as sequências quanto no comprimento destas sequências. E neste caso, o comprimento destas sequências podem variar até um comprimento máximo N. Na verdade, no algoritmo genético padrão o comprimento dos indivíduos é constante. A possibilidade de comprimento variante é uma introdução feita em outras versões de algoritmos genéticos.

Outros dados de entrada são os estados inicial e final, respectivamente, dados por M_{0i} e M_g, vetores cujos componentes contêm o número de *tokens* em cada *place* discreto p_k, para $k = 0, \ldots, NPd$, onde NPd é o número de *places* discretos. Além disso, são também necessários o tamanho da população, M, o número de gerações, G, e as probabilidades de cruzamento, mutação e reprodução, p_c, p_m e p_r, respectivamente.

População Inicial

Uma sequência s_i tem o formato geral descrito na figura 6.1. Os valores Nt_j para $j = 1, \ldots, NPr$, onde NPr é o número de subprocessos, representam o comprimento da sub-

sequência de transições referentes ao subprocesso j. Nesta subsequência a seleção das transições ocorrem em TD_j, onde TD_j representa o conjunto das transições discretas do subprocesso j. A determinação dos valores de Nt_j para cada sequência é feita aleatoriamente[1] no intervalo

$$[0, \lfloor (N - NPr)/NPr \rfloor],$$

onde, N é o comprimento máximo permitido para as sequências. O limite superior deste intervalo, representa o número médio de transições em uma subsequência em relação ao comprimento máximo da sequência e ao número de subprocessos.

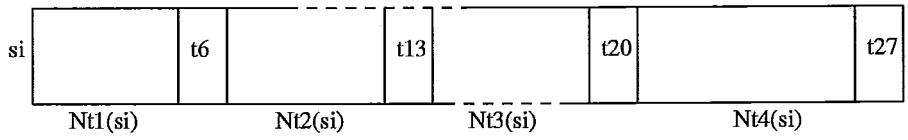

Figura 6.1: Formato geral de uma sequência

Desta forma, permite-se que as subsequências relativas aos subprocessos tenham comprimentos proporcionais, não privilegiando um subprocesso em relação a outro. A seleção aleatória de Nt_j visa a garantir a diversidade de comprimentos de sequências na população. Desta maneira é possível haver os diferentes comprimentos no intervalo $[NPr, N]$. As subsequências com comprimentos maiores que N são desprezadas tanto na geração quanto na evolução do algoritmo.

Observe que o menor comprimento possível é NPr. De fato, como pode ser visto, as transições que indicam fim de subprocesso (conforme figura 6.1) são as transições t_6, t_{13}, t_{20} e t_{27}, que estão presentes em toda sequência. Sendo assim, a menor sequência que pode ser

[1] O algoritmo para geração de números aleatórios está baseado em Park[39].

obtida é $t_6\ t_{13}\ t_{20}\ t_{27}$ e neste caso, $Nt_j = 0$ para todo $j = 1, \ldots, NPr$. A seguir tem-se o algoritmo para a geração da população.

Passo 1: Determinar $NT = \lfloor (N - NPr)/NPr \rfloor$, o número máximo de transições para os subprocessos

Passo 2: Selecione nt, o número de transições para o subprocesso entre 0 e NT

Passo 3: Selecione nt transições no conjunto de transições discretas relativas ao subprocesso

Passo 4: Acrescente a transição de final de subprocesso

Vá ao passo 2 até que NPr

Alguns exemplos de sequências são dados abaixo:

1. $s_1 : t_6 t_{13}\ t_{19} t_{17}\ t_{20}\ t_{23} t_{26}\ t_{27}$, com $Nt_1 = Nt_2 = 0$, $Nt_3 = Nt_4 = 2$

2. $s_2 : t_0 t_5\ t_6\ t_{11} t_9\ t_{13}\ t_{20}\ t_{23} t_{26}\ t_{27}$, com $Nt_1 = Nt_2 = 2$, $Nt_3 = 0$ e $Nt_4 = 2$

3. $s_3 : t_4 t_3\ t_6\ t_{10} t_8\ t_{13}\ t_{15} t_{18}\ t_{20} t_{27}$, com $Nt_1 = Nt_2 = Nt_3 = 2$ e $Nt_4 = 0$

Diz-se então que as sequências representam planos de ação. Assim, a execução de s_1, da esquerda para a direita, diz que t_6 é disparada em M_0, gerando M_1, no qual se dispara t_{19}, gerando M_2, e assim por diante, até que se tenha M_{f_1}, o estado final gerado por s_1, depois do disparo de t_{26}. Observe que as transições de fim de subprocesso não são disparadas. De fato, estas apenas representam o fim dos subprocessos. O principal objetivo destas transições é possibilitar a supervisão, pois elas são pontos para identificação de problemas.

Função de Adaptação

Desde que as sequências são aleatoriamente geradas é necessário verificar a viabilidade de disparar a primeira transição, $t_{inicial}$, no estado inicial.

Primeiro Critério: Determinar se a primeira transição está capacitada no estado inicial, através da regra:

$$M_0(p_k) \geq Pre(p_k, t_{inicial}) \quad \forall p_k \in PD$$

onde, PD é o conjunto de *places* discretos.

Para cada sequência disparável no estado inicial, determina-se o estado final atingido por esta sequência, M_f, através da regra de funcionamento, $M_f = M_0 + W.S$, onde S é o vetor das transições, M_0 o estado inicial e W a matriz dada por $Pos - Pre$, relativa apenas às transições discretas. Considera-se também, M_g, o estado final a ser alcançado.

Segundo Critério: Avaliação da distância entre M_f e M_g, através da seguinte função:

$$f(M_f) = \sum_{i=0}^{NPd-1} |M_f(p_i) - M_g(p_i)|^2$$

Observe que $f(M_f)$ determina as diferenças entre os estados final gerado pela sequência e o estado final desejado.

Note que o máximo valor de f é NPd, e neste caso todos os componentes de M_f e M_g são diferentes, ou seja, a distância entre M_f e M_g é máxima. E assim, a sequência de transições que resulta em M_f não é considerada uma solução para o problema. As sequências em que o primeiro critério não se verifica recebem este máximo valor da função f e não são avaliadas pelo segundo critério.

Assim, a função de adaptação f_{adapt} é dada por:

$$f_{adapt}(s) = \begin{cases} NPd, & \text{se } \exists \ p_k \in PD \text{ tal que } M_0(p_k) < Pre(p_k, t_{inicial}) \\ f(M_f), & \text{caso contrário.} \end{cases}$$

Observe que o mínimo desta função é zero, que ocorre quando $M_f = M_g$, ou seja, M_f é o estado final desejado e a sequência de transições que o gera é possivelmente uma sequência ótima. Assim, o problema pode ser resumido em

$$min f_{adapt}$$

Operadores

Os operadores propostos na versão original são utilizados, ou seja, os operadores cruzamento, mutação e reprodução.

1. Cruzamento

   Como as sequências são geradas levando em consideração os subprocessos, um só ponto de cruzamento não é suficiente, desde que isto poderia destruir a subdivisão estabelecida nas sequências. Assim, baseado neste fato o número de pontos de cruzamento é determinado de acordo com o número de subprocessos. Então, dado que existem NPr subprocessos, existem P_i pontos de cruzamento para $i = 1, \ldots, NPr$, isto é, um ponto em cada subsequência relativa a um subprocesso.

   Dado que as subsequências podem ter comprimentos diferentes, é necessário analisar em quais das duas sequências aleatoriamente selecionadas para cruzamento, serão selecionados os pontos de cruzamento.

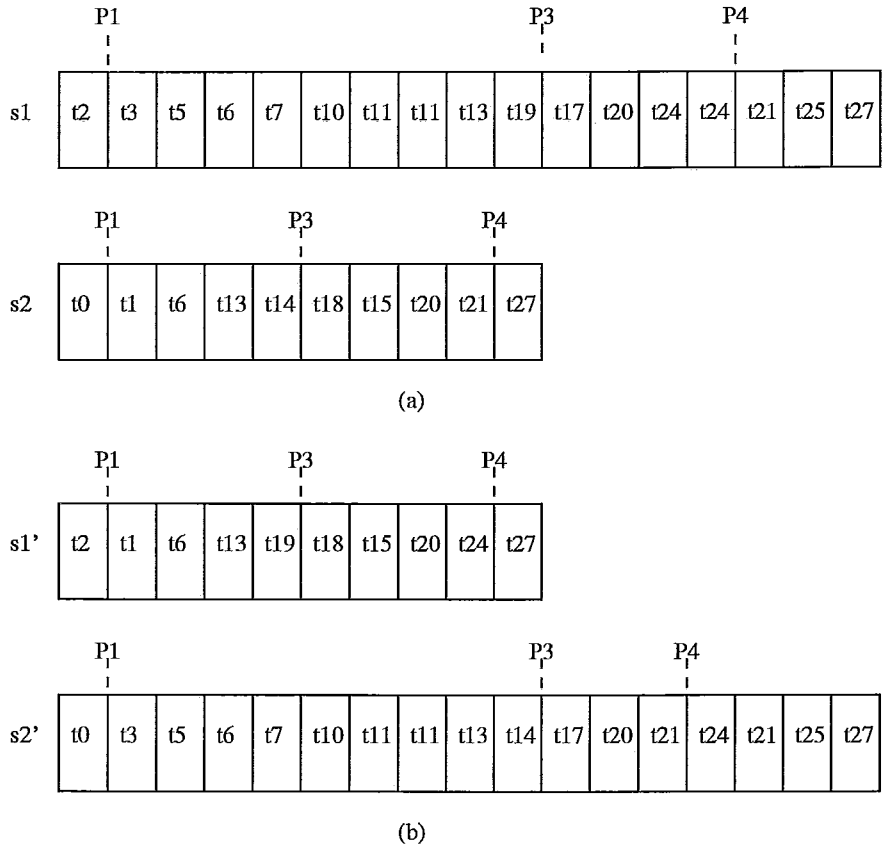

Figura 6.2: O operador cruzamento

Assim, as duas sequências s_1 e s_2 na figura 6.2 (a), após cruzamento formarão duas novas sequências, $s_1{}'$ e $s_2{}'$ (figura 6.2 (b)). A seleção de pontos de cruzamento é feita através da comparação dos $Nt_j(s_1)$ e $Nt_j(s_2)$ para $j = 1, \ldots, NPr$. Observe que o ponto de cruzamento deve ser selecionado na sequência em que Nt_j é menor, pois se fosse selecionado na sequência em que NT_j é maior, haveria a possibilidade deste ponto não estar presente na sequência menor. Os pontos selecionados foram P_1, P_3 e P_4, como indicado na figura 6.2.

Para o subprocesso 1, por exemplo, observe que $Nt_1(s_1) = 3$, $Nt_1(s_2) = 2$ e portanto, $Nt_1(s_2) < Nt_1(s_1)$. Então, o ponto de cruzamento P_1 para a primeira subsequência é selecionado aleatoriamente na sequência s_2. Como $Nt_1(s_2) = 2$, P_1 é tomado entre as posições 1 e 2, suponha então que $P_1 = 1$. Assim, pode-se realizar a primeira troca entre as duas sequências, s_1 e s_2, considerando apenas as transições existentes entre a posição $P_1 + 1$ e a posição da transição de fim do subprocesso 1.

Observe que o ponto de cruzamento P_2 não é determinado, pois neste caso, $Nt_2(s_2) = 0$, ou seja, não existem transições relativas ao subprocesso 2 na sequência s_2. Assim, apenas copia-se as transições de s_1 para s_2 relativas a este subprocesso. $P_3 = 2$ e poderia ter sido selecionado em quaisquer das sequências, dado que $Nt_3(s_1) = Nt_3(s_2)$. Por fim, $P_4 = 1$ e foi selecionado em s_2, dado que $Nt_4(s_2) < Nt_4(s_1)$. Neste caso, nenhuma transição de s_2 é copiada em s_1, pois a partir da posição 2 não existem transições para o subprocesso 4 em s_2. Um algoritmo para este operador é dado abaixo, onde p_c é a probabilidade de cruzamento.

Passo 1: Determinar $CS = \{s_1, \ldots, s_{p_c \times M}\}$, o conjunto de sequências que podem ser selecionadas para cruzamento

Passo 2: Selecione aleatoriamente s_i e s_j em CS

Passo 3: Selecione pontos de cruzamento P_k, com $k = 1, \ldots, NPr$

Passo 4: Troque as subsequências de s_i e s_j de acordo com os pontos P_k

2. Mutação

Este operador é semelhante ao operador cruzamento no aspecto da seleção de pontos para mutação em cada subsequência. Assim, aleatoriamente, uma sequência s_k é selecionada e também posições P_i para $i = 1, \ldots, NPr$ são selecionadas em s_k, de acordo com $Nt_i(s_k)$. Daí, muta-se a transição existente nestas posições P_i por outras selecionadas aleatoriamente nos conjuntos PD_i.

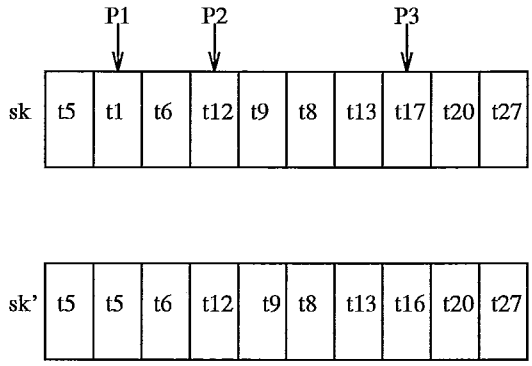

Figura 6.3: O operador mutação

Na figura 6.3 tem-se um exemplo da aplicação deste operador à sequência s_k, onde $P_1 = 2$, $P_2 = 1$, $P_3 = 1$ e $P_4 = 0$ e o resultado é a sequência $s_k{}'$, onde em P_1 a transição t_1 foi substituída por t_5, em P_2 t_{12} foi substituída por ela mesma e em P_3

t_{17} foi substituída por t_{16}. De fato, não se exclui a transição já existente da fase de seleção de uma outra transição para substituição. Este caso de mutação pode ser visto no subprocesso 2 da figura 6.3, onde a transição t_{12} foi substituída por ela mesma. O algoritmo que se segue apresenta os passos relativos a este operador.

Passo 1: Selecione a sequência s_i na população atual

Passo 2: Selecione pontos de mutação P_k, com $k = 1, \ldots, NPr$

Passo 3: Selecione aleatoriamente uma transição discreta t_j em TD_k

Passo 4: Substitua a transição em P_k por t_j

3. Reprodução

Toma-se apenas a melhor sequência da população atual e a copia na população seguinte.

Ainda a respeito das probabilidades, tem-se que um indivíduo de cada população tem sua origem no operador reprodução, os demais $M - 1$ são formados a partir da seleção de indivíduos para cruzamento entre os pontos $p_c \times M$ melhores e em $p_m \times M$ indivíduos ocorre mutação.

Observe que a interpretação para a probabilidade de cruzamento não é a porcentagem da população seguinte que é formada a partir de cruzamento. p_c significa a porcentagem de indivíduos da população anterior que poderão ser selecionados para cruzamento. Tal interpretação pode ser vista em Koza[26, 27]. Assim, o primeiro algoritmo implementado para a parte discreta é descrito abaixo[2].

[2]Todos os algoritmos neste trabalho foram implementados utilizando a linguagem C do ambiente UNIX.

Passo 1: Gere população inicial, g_0

Passo 2: Avalie cada sequência em g_0 através de f_{adapt}

Passo 3: Ordene sequências em g_0, de forma decrescente, de acordo com f_{adapt}

Passo 4: Gere população seguinte, g

Passo 4.1: Gere $M-1$ sequências através de cruzamento

Passo 4.2: Gere uma sequência através do operador reprodução

Passo 5: Faça mutação em $p_m \times M$ sequências de g

$g_0 = g$

Vá ao passo 2 até que G

Resultados

A tabela 6.1 mostra os resultados para cinco estados iniciais dados por M_{01}, ..., M_{05} e um único estado final desejado dado por M_g. Apenas este estado está sendo considerado pois é aquele que representa o estado em que a parte discreta da planta permite o início da operação da parte contínua. Além disso, como se pretende realizar o "start-up" da planta, é neste estado que tal situação está representada. Assim, apenas os estados iniciais variaram, ou seja, apenas variou-se a situação em que se encontrava a planta antes da inicialização do processo.

Os demais dados presentes na tabela 6.1 são G, o número de gerações (ou ainda, o número de passos que o algoritmo foi executado), M, o tamanho da população, N o comprimento máximo das sequências, p_c, a probabilidade do operador cruzamento e p_m, a probabilidade

do operador mutação. As sequências finais obtidas em relação a cada estado inicial, M_{0i} com $i = 0, \ldots, 5$, são indicadas por s_{f_i}.

Estado Iinicial	Estado Final	G	M	N	P_c	P_m	Seq. Final
M_{01}	M_g	65	140	18	0.82	0.02	s_{f_1}
M_{02}	M_g	65	140	18	0.75	0.01	s_{f_2}
M_{03}	M_g	65	140	16	0.7	0.01	s_{f_3}
M_{04}	M_g	65	140	16	0.8	0.01	s_{f_4}
M_{05}	M_g	65	140	17	0.75	0.01	s_{f_5}

Tabela 6.1: Resultados com Algoritmo Genético Padrão

onde, os estados iniciais M_{0i} e o estado desejado M_g são descritos na tabela 6.2. $PD = \{p_0, \ldots, p_{27}\}$ indica o conjunto de *places* discretos. Assim, a coluna relativa a cada M_{0i} e a M_g indica o número de *tokens* em *place* k, com $k = 0, \ldots, 27$. E os s_{f_i} são dados por:

$s_{f_1} : t_4t_0t_2t_6t_9t_7t_{11}t_{13}t_{16}t_{18}t_{14}t_{18}t_{20}t_{21}t_{23}t_{25}t_{27}$

$s_{f_2} : t_3t_2t_6t_{11}t_9t_{13}t_{20}t_{25}t_{21}t_{23}t_{27}$

$s_{f_3} : t_4t_2t_0t_6t_{13}t_{17}t_{16}t_{20}t_{22}t_{21}t_{26}t_{25}t_{27}$

$s_{f_4} : t_4t_0t_6t_{13}t_{18}t_{20}t_{25}t_{27}$

$s_{f_5} : t_0t_4t_6t_7t_{13}t_{14}t_{18}t_{19}t_{16}t_{18}t_{20}t_{21}t_{25}t_{21}t_{27}$

A seleção das sequências para cruzamento é feita aleatoriamente entre as $p_c \times M$ "melhores" sequências, onde "melhor" é entendido como a que tem menor valor de função de adaptação, desde que se deseja uma minimização. Para estes exemplos, $M = 140$ é o tamanho da população, e considerando $p_c = 0.8$ e $p_m = 0.02$, tem-se então que na aplicação do operador cruzamento, os indivíduos foram selecionados entre os 112 indivíduos melhores

PD	M_{01}	M_{02}	M_{03}	M_{04}	M_{05}	M_g
p_0	0	1	0	0	0	1
p_1	0	1	0	1	1	1
p_2	0	1	0	0	0	1
p_3	1	0	1	1	1	0
p_4	1	0	1	0	0	0
p_5	1	0	1	1	1	0
p_6	0	0	0	0	0	0
p_7	1	1	0	1	1	0
p_8	1	1	0	0	0	0
p_9	1	1	0	0	0	0
p_{10}	0	0	1	1	1	1
p_{11}	0	0	1	1	1	1
p_{12}	0	0	1	0	0	1
p_{13}	0	0	0	0	0	0
p_{14}	1	0	0	0	1	0
p_{15}	1	0	0	0	1	0
p_{16}	1	0	0	1	1	0
p_{17}	0	1	1	0	0	1
p_{18}	0	1	1	1	0	1
p_{19}	0	1	1	1	0	1
p_{20}	0	0	0	0	0	0
p_{21}	1	1	0	0	1	0
p_{22}	1	1	0	0	0	0
p_{23}	1	1	0	1	1	0
p_{24}	0	0	1	0	0	1
p_{25}	0	0	1	1	1	1
p_{26}	0	0	1	1	0	1
p_{27}	0	0	0	0	0	0

Tabela 6.2: Dados de entrada

adaptados e assim foram gerados os 139 restantes indivíduos para a população seguinte. Entre os 140 antigos indivíduos apenas 2 foram selecionados para mutação.

As sequências finais ótimas são dadas por $s_{f_1}, \ldots, s_{f_5}$. Os principais aspectos a serem analisados em tais sequências estão relacionados à obtenção de M_g e a existência de ciclos.

1. Em s_{f_1}, todas as transições são as necessárias, mas no subprocesso 3, há a repetição da transição t_{18}.

2. Em s_{f_2}, a subsequência $t_3\ t_2$ no subprocesso 1 é um ciclo. No subprocesso 2 a transição t_7 deveria estar presente.

3. Em s_{f_3}, existem três ciclos, $t_{17}\ t_{16}$, $t_{22}\ t_{21}$ e $t_{26}\ t_{25}$. As demais são as transições necessárias.

4. Em s_{f_4}, para o subprocesso 2 era necessário a presença da transição t_7, para os demais não há ciclos e estão presentes apenas as transições necessárias e sem repetições.

5. Em s_{f_5}, há o ciclo $t_{18}\ t_{19}$ e a repetição de t_{21} nos subprocessos 3 e 4, respectivamente, as demais são transições necessárias.

A existência de ciclos não impede que se atinja o estado final desejado. Entretanto, a repetição e a ausência de transições necessárias não conduzem ao estado desejado. Mas, observe que a ocorrência destes dois casos não é grande e considerando que através de algoritmos genéticos obtém-se uma solução ótima ou próxima do ótimo, pode-se dizer que estas soluções são ótimas no sentido que as diferenças entre os estados M_f e M_g são pequenas.

Mas, analisando sob o aspecto de tornar tal procedimento para "start-up" automático, ou seja, considerando a não existência do operador humano, tanto as repetições quanto os ciclos e as ausências de transições representam problemas. Entretanto, o objetivo é oferecer um auxílio ao operador humano e daí, pelo menos as repetições e os ciclos poderiam ser resolvidos através de uma análise dos resultados. Mas a ausência de transições também neste caso é um problema.

Observe que alguns parâmetros como G e M, que representam o número de gerações e o tamanho da população, respectivamente, permaneceram os mesmos para estes cinco exemplos. De fato, tentou-se estabilizar os parâmetros que são considerados os princiapis responsáveis pela complexidade de tempo em algoritmos genéticos. Contudo, os demais parâmetros não sofreram grandes variações.

6.1.2 Proposta de um Algoritmo Genético Melhorado

Como dito na introdução deste capítulo, nesta versão do algoritmo a população inicial é gerada a partir da gramática associada à Rede de Petri. De forma resumida, nesta versão do algoritmo, cada sequência é um caminho na árvore de cobertura que pode ser construída a partir de um estado inicial M_0. Será visto, a seguir, cada um dos elementos que definem o algoritmo, população inicial, função de adaptação e operadores.

Será observado que devido à forma de geração da população inicial adotada nesta versão, os outros elementos do algoritmo, função de adaptação e operadores serão modificados. Isto é, estes elementos como definidos na versão padrão do algoritmo não são adequados para este caso. Assim, quanto à função, esta é simplificada. E quanto aos operadores, introduz-se

um novo operador genético que preserva a forma de geração das sequências.

População Inicial

Na formação das sequências para a população inicial, neste caso, observa-se os fatos de que todas as sequências geradas são disparáveis no estado inicial e toda transição é disparável no estado gerado pela transição anterior. Assim, a cada nova transição que se acrescenta à sequência em formação é necessário levar em consideração o estado gerado por esta sequência. Isto é, suponha que para o estado inicial M_0 dado uma transição t_i capacitada, ou seja, $M_0(p_j) \geq Pre(p_j, t_i) \quad \forall p_j \in PD$.

Do disparo de t_i em M_0 tem-se um novo estado, M_1. Assim, para se gerar uma nova transição para sequência repete-se o processo anterior, tomando em lugar de M_0 o estado M_1, que neste caso poderia ser visto como o estado inicial da sequência em formação. Note, entretanto que para cada estado pode haver mais do que uma transição capacitada, ou seja, pode haver um conjunto de transições capacitadas, mas apenas uma deve ser tomada a cada instante. Sendo assim é necessário estabelecer um critério de seleção. O critério foi a seleção aleatória. Assim, dentro do conjunto de transições capacitadas num dado estado, toma-se uma aleatoriamente.

Os comprimentos das sequências e das subsequências são determinados como anteriormente, ou seja, os valores de NTj e o comprimento máximo das sequências, N, são todos obtidos com no caso do algoritmo genético padrão. Desta forma de geração tem-se que todas as sequências geradas fazem parte da árvore de cobertura da rede de Petri em relação à parte

discreta. Sendo assim, sabe-se que a população inicial é constituída apenas de sequências disparáveis em relação à rede de Petri.

Note que apesar de ser gerada de acordo com a gramática, apenas se considera as regras relativas à parte discreta da rede, ou seja, apenas as transições discretas. As regras que relacionam *places* discretos e contínuos se referem às transições contínuas ou às transições que determinam o fim de um subprocesso. Assim, o algoritmo para geração da população inicial de acordo com a gramática da rede de Petri híbrida é dado a seguir.

Passo 1: Determinar $NT = \lfloor (N - NPr)/NPr \rfloor$, o número máximo
de transições para os subprocessos

Passo 2: Selecione nt, o número de transições para o
subprocesso entre 0 e NT

Passo 3: Determine T, o conjunto de transições
disparávies em M_0

Passo 4: Selecione aleatoriamente uma transição t em T

Passo 5: Dispare t em M_0 e gere novo estado M

Passo 6: $M_0 = M$
Vá ao Passo 3 até que nt

Passo 7: Acrescente a transição de fim de subprocesso
Vá ao passo 2 até que NPr

Função de Adaptação

Dada que todas as sequências geradas estão de acordo com a gramática associada à rede de Petri tem-se que a primeira transição na sequência pode ser disparada no estado inicial dado, logo o primeiro critério utilizado no caso da geração aletória não se faz necessário, pois todas as sequências geradas podem ser disparadas no estado inicial. Apenas o segundo critério deve ser levado em consideração. Sendo assim, a função de adaptação para este caso é dada por:

$$f_{adapt}(s_i) = f(M_{f_i})$$

e o problema consiste em:

$$min\ f_{adapt}$$

Operadores

Os operadores genéticos padrão não podem ser utilizados com esta forma de geração, dado que estes poderiam destruir as sequências previamente geradas, segundo a gramática, na população inicial. Por isso, um novo operador, denominado mutação-cruzada é introduzido, e como consiste em uma mistura de mutação e cruzamento, apenas este e a reprodução são utilizados nesta versão do algoritmo.

1. Mutação-Cruzada

   Cruza-se os comprimentos das subsequências e por consequência os comprimentos de s_1 e s_2, como no algoritmo padrão, e então muta-se partes das subsequências através da geração de novas transições a partir dos P_i pontos de cruzamento. Desta

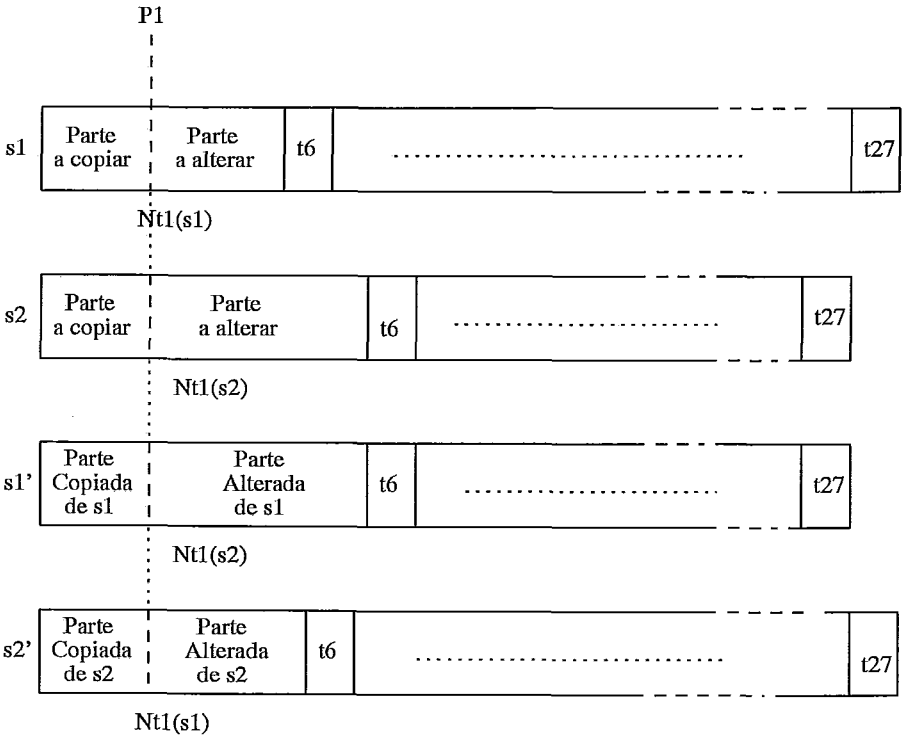

Figura 6.4: Operador Mutação-Cruzada

maneira garante-se que as sequências continuem sendo sequências disparáveis durante as gerações seguintes.

Além disso, um fato relevante é que este operador permite a introdução de novas composições de transições em cada sequência de forma a garantir a diversidade das populações. De fato, manter a diversidade durante a evolução do algoritmo é um fato importante, dado que argumentações mostram que o cruzamento embora seja o principal operador em algoritmos ele também pode provocar a não convergência ou a convergência prematura (para pontos não desejados) do algoritmo, dado que ele pode especializar em excesso as populações. Mas, este novo operador introduzido evita que tal fato ocorra devido à introdução de novas transições. Os passos para aplicação deste operador é dado a seguir.

Passo 1: Determinar $CS = \{s_1, \ldots, s_{p_c \times M}\}$, o conjunto de sequências que podem ser selecionadas para cruzamento

Passo 2: Selecione aleatoriamente s_i e s_j em CS

Passo 3: Selecione pontos para mutação-cruzada P_k, com $k = 1, \ldots, NPr$

Passo 4: Para cada subprocesso troque as subsequências de s_i e s_j de acordo com os pontos P_k

Passo 5: A partir do estado gerado nos pontos P_k, gere novas transições para s_i de acordo com o comprimento de s_j

Passo 6: A partir do estado gerado nos pontos P_k, gere novas

transições para s_j de acordo com o comprimento de s_i

2. Reprodução

   Toma-se apenas o melhor indivíduo de cada população e o copia na população seguinte, como no caso anterior do algoritmo genético padrão.

Assim, a versão do algoritmo baseado nestes operadores foi implementada de acordo com os passos abaixo.

Passo 1: Gere população inicial, g_0

Passo 2: Avalie cada sequência em g_0 através de f_{adapt}

Passo 3: Ordene sequências em g_0, de forma decrescente, de acordo com f_{adapt}

Passo 4: Gere população seguinte, g

Passo 4.1: Gere $M-1$ sequências através de mutação-cruzada

Passo 4.2: Gere uma sequência através do operador reprodução

$g_0 = g$

Passo 5: Vá ao passo 2 até que G

Resultados

A tabela 6.3 descreve os resultados da aplicação do algoritmo melhorado aos dados de entrada apresentados na tabela 6.2, onde G é o número de gerações, M é o tamanho da população, N é o comprimento máximo das sequências e p_c a probabilidade para o operador mutação-cruzada. E os s_{f_i} são dados por:

s_{f_1} : $t_2t_0t_4t_6t_7t_8t_7t_9t_{11}t_{13}t_{14}t_{18}t_{16}t_{20}t_{21}t_{25}t_{23}t_{27}$

s_{f_2} : $t_6t_9t_{11}t_7t_{13}t_{20}t_{21}t_{23}t_{25}t_{27}$

s_{f_3} : $t_4t_0t_2t_6t_{13}t_{20}t_{27}$

s_{f_4} : $t_4t_0t_6t_7t_{13}t_{17}t_{16}t_{18}t_{20}t_{22}t_{25}t_{21}t_{27}$

s_{f_5} : $t_0t_4t_6t_{10}t_7t_9t_{13}t_{16}t_{18}t_{14}t_{20}t_{21}t_{25}t_{26}t_{25}t_{27}$

Estado Iinicial	Estado Final	G	M	N	P_c	Seq. Final
M_{01}	M_g	45	100	20	0.8	s_{f_1}
M_{02}	M_g	40	100	13	0.8	s_{f_2}
M_{03}	M_g	30	100	10	0.8	s_{f_3}
M_{04}	M_g	20	50	15	0.8	s_{f_4}
M_{05}	M_g	45	100	19	0.75	s_{f_5}

Tabela 6.3: Resultados com Algoritmo Genético Melhorado

1. Em s_{f_1}, há o ciclo t_7 t_8 no subprocesso 1, as demais são as transições necessárias.

2. Em s_{f_2}, não há ciclos e as transições são as necessárias.

3. Em s_{f_3}, não há ciclos e as transições são as necessárias.

4. Em s_{f_4}, há dois ciclos, t_{17} t_{16} e t_{22} t_{21}, e as demais são transições necessárias.

5. Em s_{f_5}, há ciclos, t_{10} t_9 e t_{26} t_{25}, e as demais são transições necessárias.

Observe que nestes resultados há apenas a presença de ciclos, sem repetições e sem ausências de transições. E sendo assim, o estado final desejado é atingido por todas as sequências. Observe que s_{f_2} e s_{f_3} são as soluções exatas. Além disso, comparando as duas tabelas de resultados, pode-se ver que os parâmetros G e M, diminuem no segundo algoritmo

e até alguns valores de N são menores. A probabilidade de cruzamento é praticamente a mesma para os dois algoritmos.

6.2 Busca e Otimização da Parte Contínua

Como visto numa rede de Petri híbrida, as transições contínuas representam eventos contínuos. No exemplo da figura 4.10, cada uma das transições está relacionada ao evento contínuo de fluxo de substâncias das fontes para os permutadores e destes para os sumidouros. Entretanto, algumas restrições quanto às temperaturas e às quantidades das substâncias nos sumidouros devem ser respeitadas. E para que isto ocorra é necessário controlar a entrada dos dois fluxos, quente e frio, em cada permutador. Tal controle é feito de acordo com as relações entre as quantidades e temperaturas dos fluxos de entrada e as quantidades e temperaturas dos fluxos de saída dadas por um sistema de equações diferenciais, conforme descrito em Fusillo[11].

$$\frac{dT^C}{dA} = \frac{U}{F^C.C_p^C}.(T^H - T^C)$$

$$\frac{dT^H}{dA} = -\frac{U}{F^H.C_p^H}.(T^H - T^C)$$

com limites

$T^C = T_{in}^C$, a temperatura de entrada do fluxo frio

$T^H = T_{in}^H$, a temperatura de entrada do fluxo quente

onde,

$$U = \left[\frac{1}{h^H_{proj}} \left(\frac{F^H_{proj}}{F^H} \right)^{0.8} + \frac{1}{h^C_{proj}} \left(\frac{F^C_{proj}}{F^C} \right)^{0.8} \right]^{-1} \text{ com } h \cong F^{0.8}$$

e A é a área do permutador, F^C e F^H são as taxas dos fluxos frio e fluxo quente, respectivamente.

O coeficiente U relaciona as taxas de fluxo reais e de projeto dos dois fluxos que entram em cada permutador. E o sistema de equações relaciona a variação da temperatura de cada fluxo, de entrada e de saída, com a taxa de fluxo. Como entram dois fluxos com temperaturas diferentes, também saem dois fluxos, um frio e outro quente. Assim, a primeira equação indica a variação de temperatura entre os fluxos frios e a segunda entre os fluxos quentes. Os coeficientes C_p e h são dados. $C_p = 0.5$[3] para todo tipo de fluxo. $h = 300$ para fluxos líquidos e $h = 1000$ para fluxos em condensação.

Na tabela 6.4 tem-se os dados de entrada para o problema, que também são apresentados na planta(figura 4.9).

Fluxo	Temperatura	Permutador	Área
H_1	400	$HX1$	33.3
C_1	100	$HX2$	84.8
H_2	275	$HX3$	13.2
C_2	125	$HX4$	79.6

Tabela 6.4: Dados de Entrada

Além destes dados, a tabela 6.5 apresenta os valores de algumas variáveis do problema no estado estável.

[3]Apenas por simplicidade de notação foram omitidas as unidades das variáveis.

Estado Estável			
Fluxo	Taxa de Fluxo	Fluxo	Temperatura
H_1	20000	$H_1 1$	300
H_2	60000	$H_2 2$	250
C_1	20000	$C_1 2$	250
C_2	80000	$C_2 2$	175

Tabela 6.5: Variáveis no Estado Estável

Por fim, na tabela 6.6 tem-se os valores de algumas variáveis no estado "goal", ou seja, valores que deveriam ocorrer no estado final desejado.

Estado "Goal"		
Fluxo	Taxa de Fluxo	Temperatura
H_{11}	20000	300
H_{22}	60000	250
C_{12}	20000	250
C_{22}	80000	175

Tabela 6.6: Variáveis no Estado Goal

Além destas tem-se que as restrições são dadas por:

1. $285 \leq T_{H11} \leq 320$, para garantir condições adequadas de reação.

2. $T \geq 150$, para garantir a solubilidade de uma das substâncias presentes no fluxo $C22$.

3. $F_{C12} = 5000 \pm 500$, pois este fluxo é usado em um reator.

Assim, nesta parte do algoritmo pretende-se determinar algumas taxas de fluxo de maneira que se produza os fluxos de saída nas quantidades e temperaturas desejadas, como descrito no estado "goal".

Para tanto, utiliza-se um algoritmo baseado em Homaifar et al.[37] e Fogel[38], que introduziram um algoritmo genético para otimização com restrições. As principais características deste algoritmo são a presença de uma função de penalidade, desde que se tem uma otimização com restrições, e a representação de cada indivíduo na população dada por uma cadeia de valores reais, ou seja, cada um dos componentes de uma cadeia está associado a uma variável do problema.

Como anteriormente feito para os algoritmos da parte discreta do problema, descreve-se a seguir cada um dos passos necessários para construção do algoritmo relativo à parte contínua do problema. Isto é, as descrições da geração da população inicial, da função de adaptação e dos operadores considerados.

População Inicial

Considera-se um indivíduo como uma cadeia de valores reais associados a uma variável do problema. Assim, o formato geral de uma sequência é dado por:

$$x_0\ x_1\ \ldots\ x_n, \text{ onde } x_i \in \Re$$

Para formação de uma sequência, seleciona-se aleatoriamente um valor para cada variável na cadeia, de acordo com as restrições dadas por desigualdades que identificam na verdade os intervalos de variação das variáveis.

No exemplo as variáveis são as transições contínuas $tc_0 \ldots tc_{11}$, que representam taxas de fluxo. Então, uma cadeia tem o seguinte formato:

$$tc_0\ tc_1\ tc_2\ tc_3\ tc_4\ tc_5\ tc_6\ tc_7\ tc_8\ tc_9\ tc_{10}\ tc_{11}$$

que representam as 12 transições contínuas da rede de Petri da figura 4.11.

As restrições para cada uma destas transições estão baseados nas tabela 6.5 e tabela 6.6, que contém as taxas de fluxo que devem ocorrer ou no estado estável e também no estado "goal" (o estado desejado) do sistema. Assim, todas as taxas de fluxo desejadas nestes dois estados são novamente apresentadas na tabela 6.7.

Estado "Goal"		Estado Estável	
Fluxo	Taxa de Fluxo	Fluxo	Taxa de Fluxo
H_{11}	20000	H_1	20000
H_{22}	60000	H_2	60000
C_{12}	5000	C_1	20000
C_{22}	80000	C_2	80000

Tabela 6.7: Taxas de fluxo nos estados estável e "goal"

Assim, tomou-se as restrições como abaixo:

$$19000 \leq tc_0,\ tc_1,\ tc_2,\ tc_7,\ tc_8 \leq 21000$$

$$59000 \leq tc_3,\ tc_4,\ tc_{10} \leq 61000$$

$$79000 \leq tc_5,\ tc_6,\ tc_{11} \leq 81000$$

$$4900 \leq tc_9 \leq 5100$$

Estes intervalos foram assim tomados, baseado no fato de que se tem o valor a ser atingido, logo permitiu-se que as variáveis variassem em torno do valor desejado. Intervalos maiores poderiam ter sido considerados, entretanto, mais casos teriam de ser analisados quando da aplicação da função penalidade, como será visto a seguir.

Portanto, a população é constituída de M cadeias com o formato acima e sendo os valores destas variáveis selecionados aleatoriamente dentro dos intervalos acima.

Função de Adaptação

Como visto na tabela 6.5, o estado "goal" e o estado estável são idênticos em relação às taxas de fluxo e em relação às temperaturas. Assim, baseado na existência destes estados, a função é definida por:

$$f(Var_{res}) = \sum_{i=1}^{n} \|Var\_goal_i - Var\_res_i\|^2$$

onde, o estado a ser atingido (estado "goal") é dado pelo vetor

$$Var\_goal = (F_{H11}, F_{C12}, F_{H22}, F_{C22}, T_{H11}, T_{C12}, T_{H22}, T_{C22})$$

e o vetor obtido pelo algoritmo é dado por:

$$Var\_res = (tc_8, tc_9, tc_{10}, tc_{11}, T_1, T_2, T_3, T_4)$$

As transições tc_8, tc_9, tc_{10} e tc_{11} representam os fluxos F_{H11}, F_{C12}, F_{H22} e F_{C22}, respectivamente. As variáveis T_1, T_2, T_3 e T_4 são os valores das temperaturas destes fluxos obtidas após simulação dos permutadores.

Uma função de penalidade para avaliar as violações das restrições impostas aos valores das transições é definida em Homafair et al.[37] como:

$$F_{penalidade}(Var_res) = \begin{cases} 0, & \text{se } LI_i \leq tc_i \leq LS_i \ \ \forall tc_i \in s \\ \sum_{i=0}^{m-1} {g_i}^2(tc_i) R_i, & \text{caso contrário} \end{cases}$$

onde, $g_i(tc_i)$ é uma função real contínua, s é a sequência de transições contínuas, m é o número de restrições, R_i é o fator de penalidade, que depende do nível de violação, LI_i e LS_i os limites inferior e superior, respectivamente, em cada restrição i.

Assim, as regras para balancear o nível de violação nos dois limites do intervalo são dadas através dos valores dos $g_i^{'}s$:

Regra 1: Limite inferior é violado

$$g_i(tc_i) = LI_i - tc_i + \textit{referência}$$

Regra 2: Limite superior é violado

$$g_i(tc_i) = tc_i - (LS_i \text{ - } \textit{referência})$$

onde, *referência* pode ser ou não um nos intervalos que representam as restrições.

Na verdade, este valor é tomado de maneira que violações de mesmo nível em relação a ambos limites sejam igualmente penalizadas. Assim, não se permite que a solução se encaminhe para um ou outro limite, dado que ambas as tendências são igualmente penalizadas. Portanto, não é necessário que este valor se encontre dentro do intervalo, basta que atinja o objetivo proposto. Para o problema exemplo, referência foi considerada o ponto médio dos intervalos. Assim para a primeira restrição,

$$19000 \leq tc_0,\ tc_1,\ tc_2,\ tc_7,\ tc_8\ \leq 21000$$

tem-se que

$$g_0(tc_0) = 19000 - tc_0 + 20000$$

$$g_0(tc_0) = tc_0 - (21000 - 20000)$$

Observe que desta maneira, tanto a violação do limite superior quanto do limite inferior, que sejam do mesmo nível, são penalizadas no mesmo grau. Ou seja, tomando $tc_0 = 22000$, $g_0(tc_0) = 21000$, e para $tc_0 = 18000$, $g_0(tc_0) = 21000$.

As transições tc_1, tc_2, tc_7 e tc_8 são tratadas como no caso acima. Para tc_3, tc_4 e tc_{10}

valem os $g_1{}'s$ abaixo,

$$g_1(tc_3) = 59000 - tc_3 + 60000$$

$$g_1(tc_3) = tc_3 - (61000 - 60000)$$

Para tc_5, tc_6 e tc_{11}, tem-se

$$g_2(tc_5) = 79000 - tc_5 + 80000$$

$$g_2(tc_5) = tc_5 - (81000 - 80000)$$

E para tc_9

$$g_3(tc_9) = 4900 - tc_9 + 5000$$

$$g_3(tc_9) = tc_9 - (5100 - 5000)$$

Os valores de tc_0 dados anteriormente, $tc_0 = 22000$ e $tc_0 = 18000$, poderiam ter o mesmo fator de penalidade, R_0. De fato, define-se subintervalos acima do limite superior e abaixo do limite inferior, ou ainda, níveis de violação, nos quais estes $R_i{}'s$ são aplicados, por exemplo, para valores de violação nos intervalos, $21000 < tc_0 \leq 26000$ e $14000 \leq tc_0 < 19000$, aplica-se R_0.

A determinação deste $R_i{}'s$ é feita de forma empírica, ou seja, através de testes até se encontrar o valor adequado. Em geral, são valores pequenos, e não necessariamente é o mesmo para todos os níveis de violação. Entretanto, tentou-se aqui estabelecer o mesmo R_i para os mesmos níveis de violação. Por exemplo, para a primeira restrição, os valores de R_0 foram tomados como abaixo, onde t_{valor} é o valor de qualquer uma das transições relativas a esta restrição.

$$\text{se } t_{valor} \leq 19000 \quad R_0 = 1.75$$

$$\text{se } 19000 < t_{valor} < 21000 \quad R_0 = 0.0$$

$$\text{se } t_{valor} \geq 21000 \quad R_0 = 1.75$$

Para a segunda restrição os valores de R_1 são:

$$\text{se } t_{valor} \leq 59000 \quad R_1 = 1.25$$

$$\text{se } 59000 < t_{valor} < 61000 \quad R_1 = 0.0$$

$$\text{se } t_{valor} \geq 61000 \quad R_1 = 1.55$$

Observe que R_1 não é o mesmo para o primeiro e o terceiro nível de violação. Para a terceira restrição tem-se R_2 como abaixo:

$$\text{se } t_{valor} \leq 79000 \quad R_2 = 1.55$$

$$\text{se } 79000 < t_{valor} < 81000 \quad R_2 = 0.0$$

$$\text{se } t_{valor} \geq 81000 \quad R_2 = 1.55$$

E por fim, para quarta restrição tem-se R_3

$$\text{se } t_{valor} \leq 4900 \quad R_3 = 1.55$$

$$\text{se } 4900 < t_{valor} < 5100 \quad R_3 = 0.0$$

$$\text{se } t_{valor} \geq 4900 \quad R_3 = 1.55$$

Entretanto, existem também restrições sobre algumas variáveis, dadas pelas desigualdades abaixo, onde T_{H11} é a temperatura do fluxo $H11$, T_{C22} é a temperatura do fluxo $C22$ e F_{C12} é a taxa de fluxo de $C12$.

$$285 \leq T_{H11} \leq 320$$

$$T_{C22} \geq 150$$

$$F_{C12} = 5000 \pm 500$$

Observe contudo, que a última destas três restrições já foi considerada no conjunto de restrições anterior. De fato, foi tomada uma restrição mais forte do que esta devido à existência do valor desta variável no estado "goal".

Mas, as restrições relativas às temperaturas, só podem ser consideradas a partir do funcionamento dos permutadores de calor. E portanto, são dependentes do resultado obtido e assim não é necessário considerá-las em relação à função de penalidade.

Então para obter os valores destas temperaturas, é necessário simular o comportamento da planta para cada indivíduo gerado em cada população do algoritmo genético. Os indivíduos representam as transições contínuas e estas as taxas de fluxo de entrada e de saída dos permutadores. Assim, a simulação consiste em tomar tais taxas de fluxo e simular o comportamento de cada permutador de calor na planta, através das equações diferenciais anteriormente dadas. Resumindo, a simulação consiste em resolver as equações diferenciais para cada permutador de calor[4].

Portanto, após a simulação obtém-se entre outros valores, os valores de T_{H11} e T_{C22}. Mas, como visto os valores destas variáveis também fazem parte do estado "goal", assim como T_{H22} e T_{C12}. Assim, aplicou-se a função f nestes casos também, dado que esta função considera as diferenças entre o estado atingido e o estado desejado.

[4]Para resolução das equações utilizou-se o método dado em Press et al[40], capítulos 15 e 16.

Então, o problema pode ser definido como

$$min \quad f_{adapt}$$

onde, $f_{adapt} = f + F_{penalidade}$

Operadores

Segundo Fogel[38], o operador cruzamento pode, na verdade, destruir boas soluções que surgem durante a evolução do algoritmo, devido às trocas entre cadeias. Acreditam que estas trocas em algum grau permitem destruir soluções próximas do ótimo e também gerar outras distantes do ótimo. Sendo assim, propôs um algoritmo diferente de Homafair et al[37], que utilizam o operador cruzamento. Mas, baseado nos aspectos acima Fogel[38] utiliza em seu algoritmo apenas os operadores mutação e reprodução, assim também é o algoritmo desenvolvido nesta parte contínua do problema.

1. Mutação

   Este operador toma cada indivíduo (uma cadeia) e gera um outro pela adição de uma variável aleatória Gaussiana[5] cuja a média é zero, a cada componente da cadeia. Se o valor da função de adaptação para esta nova cadeia é melhor que da cadeia anterior, então a nova cadeia substituirá a anterior na população seguinte, em caso contrário, a cadeia anterior permanece na população seguinte. Além disso, se algum dos componentes ultrapassa o seu intervalo de variação devido ao acréscimo da variável Gaussiana, este recebe como valor o limite do intervalo violado.

[5] O método e o algoritmo foram retirados de Press et al[40], capítulo 7.

O algoritmo abaixo apresenta os passos para aplicação deste operador.

Passo 1: Tome uma sequência s_i

Passo 2: Adicione uma variável Gaussiana a cada componente de s_i, obtendo $s_i{}'$

Passo 3: Avalie $s_i{}'$

Passo 4: Compare $f_{adapt}(s_i)$ e $f_{adapt}(s_i{}')$

Passo 5: Copie na população seguinte aquela que tiver menor valor de f_{adapt}

2. Reprodução

Dada a população de tamanho M, toma-se cada indivíduo e o compara, através dos valores de função de adaptação, a C outros indivíduos selecionados aleatoriamente. O melhor entre estes $C+1$ indivíduos será copiado na população seguinte. Assim, após comparar cada um dos M indivíduos a C outros, a nova população é formada.

Desta forma, o número de cópias do melhor indivíduo na população seguinte é maior que um. Os passos que resume que tal operador são apresentados a seguir.

Passo 1: Selecione uma sequência s_i

Passo 2: Selecione um conjunto C com $p_r \times M$ sequências

Passo 3: Compare s_i com todas as sequências em C

Passo 4: Copie a melhor sequência obtida em Passo 3 na população seguinte

Assim, o algoritmo implementado para a parte contínua do problema é dado abaixo, onde G é o número de gerações.

Passo 1: Gere população inicial, g_0

Passo 2: Avalie cada sequência em g_0, de acordo com f_{adapt}

Passo 3: Gere população seguinte, g

Passo 3.1: Gere M sequências através do operador mutação

Passo 3.2: Gere M melhores sequências através do operador reprodução

Passo 4: $g_0 = g$

Vá ao Passo 2 até que G

Resultados

Fogel[38] afirma que uma população de tamanho $M = 40$ é suficiente para a evolução de um tal algoritmo. Assim, utilizou-se um população de tamanho $M = 40$, número de gerações $G = 20$ e $C = 0.5 \times M = 20$, ou seja, no operador cada indivíduo foi comparado a 20 outros.

Como nos outros algoritmos genéticos neste trabalho, o resultado considerado foi o melhor indivíduo da última geração. Assim, o resultado para a parte contínua foi a cadeia abaixo.

$$tc_0 = 19099.87\ tc_1 = 20840.10\ tc_2 = 19002.04\ tc_3 = 60277.76$$

$$tc_4 = 60119.18\ tc_5 = 80754.49\ tc_6 = 79196.80\ tc_7 = 19549.04$$

$$tc_8 = 19961.82\ tc_9 = 4971.18\ tc_{10} = 60416.19\ tc_{11} = 80669.33$$

A tabela 6.8 apresenta o estado atingido em relação às taxas de fluxo e às temperaturas desejadas no estado final. Observe que a maior diferença existente nestes resultados está na variável T_{C12}, da ordem de 27.3%. Os demais resultados estiveram próximos ao estado "goal".

	Estado "Goal"		Estado Atingido	
Fluxo	Taxa de Fluxo	Temperatura	Taxa de Fluxo	Temperatura
H_{11}	20000	300	19961.82	309.87
H_{22}	60000	250	60416.19	247.56
C_{12}	5000	250	4971.18	181.84
C_{22}	80000	175	80669.33	173.05

Tabela 6.8: Otimização Contínua

Uma análise conclusiva de todo o trabalho é apresentada a seguir, baseada nos seguintes aspectos:

1. Quanto à representação: Redes de Petri híbridas são capazes de bem representar problemas como processos químicos, a dificuldade com tal formalismo é o fato de sua evolução ser não determinística.

2. Quanto ao método de busca: A forma de evolução de redes de Petri, pode não ser, como no caso do exemplo, a menor sequência ou ainda, a sequência mais próxima da menor. Além disso, o aspecto da ordenação da sequência também pode não ser respeitado. Por fim, uma rede de Petri pesquisará uma sequência por vez. Assim, poderá primeiro pesquisar por sequências que não atingem o estado desejado e podendo ainda ser sequências de grande comprimento.

   O algoritmo genético pesquisa entre várias sequências a cada geração. Além disso, busca segundo algum critério de otimalidade e as sequências serão ordenadas, em grau de ordenação, desde que já foram assim construídas. Os ciclos ocorreram, mas com pequenos comprimentos. Mas, estes ocorreriam mesmo na própria evolução de redes

de Petri. De fato, os algoritmos genéticos devem ter permitido ciclos menores devido à existência de um critério de otimalidade.

3. Quanto aos algoritmos genéticos: Estes têm as desvantagens de um grande consumo de tempo e uma difícil sintonização de parâmetros, como tamanho da população, comprimento dos indivíduos e probabilidades. Entretanto, observa-se que em comparação a outros problemas em que foram utilizados, os parâmetros aqui obtidos nas três versões do algoritmo não são tão grandes. Além disso, o algoritmo genético proposto obteve menores valores para estes parâmetros que o algoritmo padrão, além também de apresentar melhores resultados.

4. Quanto a ser um procedimento automático: Os ciclos não foram completamente evitados, e mesmo que tenham pequenos comprimentos comprometem um possível procedimento automático para "start-up". Entretanto, quanto a um auxílio ao operador humano é viável, pois os resultados podem ser analisados antes de serem executados.

5. Quanto a um possível sistema de supervisão: Como visto no capítulo 2, um módulo para planejamento de ações é necessário num sistema de supervisão. Assim, é possível incluir tal procedimento neste esquema, onde os *places* e as transições de fim de subprocesso poderiam desempenhar o papel de ponto de recepção de informações sobre os subprocessos. Além disso, dada a divisão em subprocessos, é possível tentar identificar e resolver problemas apenas nos subprocessos para os quais estes ocorrem. Pelo menos, quanto à parte discreta. Como os subprocessos são dependentes, principal-

mente quanto à parte contínua, um problema na parte contínua de um subprocesso pode afetar os outros, envolvendo assim todos subprocessos da parte contínua. Mas, isto faz sentido desde que é nesta parte que se tem o processo propriamente dito.

Capítulo 7

Conclusões e Perspectivas Futuras

Como visto na introdução e no capítulo 1, o problema aqui considerado é o planejamento do "'start-up" de plantas químicas, cuja solução é uma sequência (plano) de ações. Assim, o objetivo do trabalho é desenvolver uma abordagem, baseada em técnicas de Inteligência Artificial, que apresentasse uma possível solução para o problema. Além disso, também foi visto no capítulo 2, que a principal dificuldade é obter um método de busca que se adapte de maneira eficiente à representação de conhecimento adotada.

Assim, na abordagem desenvolvida adotou-se redes de Petri híbridas para representação do conhecimento e algoritmos genéticos como método de busca. Neste caso, as ações são representadas pelas transições na rede de Petri. Entretanto, na rede existem várias sequências de transições que realizam o "start-up". Assim, o algoritmo genético é utilizado no sentido de tomar a sequência ótima, que é a de menor comprimento.

As vantagens de utilizar Redes de Petri é colocar todas as informações associadas numa única representação, não sendo necessário utilizar várias formas de representação como pôde ser visto nas técnias apresentadas no capítulo 2. Além disso, também as operações são representadas.

Além disso, nesta abordagem, o algoritmo foi dividido em duas partes (discreta e contínua) baseado no fato de que dentro do processo químico a parte identificada como discreta envolve operações que verificam ou determinam a possibilidade de colocar o processo propriamente dito em execução. E este, o processo é identificado com a parte contínua. Também a representação da rede e do processo se baseia neste fato a dependência da parte contínua em relação à parte discreta.

Para a parte discreta desenvolveu-se dois algoritmos genéticos: o primeiro baseado na versão padrão e o segundo introduziu um novo operador, a mutação-cruzada. Nesta parte, a solução é uma sequência de transições discretas que permitem o disparo das transições contínuas envolvidas no processo. Observou-se que, para as mesmas entradas, o segundo algoritmo obteve melhores resultados, desde que contêm menos ciclos, e por isso, são menores. Além disso, estão mais próximas da solução desejada, e alguns casos atingiu-se a solução exata.

Quanto à parte contínua, utilizou-se um algoritmo genético para otimização com restrições para determinar a quantidade de disparo de todas as transições contínuas presentes na rede. Foi visto que a solução ótima obtida estava bem próxima ao estado "goal" desejado, excetuando apenas um parâmetro. A determinação dos parâmetros deste algoritmo pode ser facilitada através da utilização de outras funções de penalidade, que evitariam a complexidade e o empirismo na determinação dos $R_i^{'}s$.

Além disso, conclui-se que apesar de se ter utilizado um problema específico, o "start-up" de plantas químicas, é possível extender esta abordagem para um problema de supervisão

de processos. Neste caso, ela seria um módulo para o planejamento de operações. Assim, tomando o estado instântaneo do processo e representando-o na rede de Petri híbrida, o algoritmo procurará por uma sequência de ações que permitirá levar o processo a um estado operacional desejado. Sendo assim, esta extensão fica como perspectivas futuras bem como a utilização de outros tipos de redes de Petri, como por exemplo, as temporizadas.

Conclui-se então, que redes de Petri híbridas e algoritmos genéticos podem compor um procedimento para o "start-up" de plantas químicas, contudo é ainda um procedimento que não pode ser automático, sendo assim, pode apenas ser visto como um auxílio ao operador humano.

Referências Bibliográficas

[1] Fari, J., Advanced Process Engineering, *AIChE Journal*, Vol., No., 1975.

[2] Rivas, J. R., Rudd, D. F. e Kelly, L. R., Computer-Aided Safety Interlock Systems, *AIChE Journal*, Vol. 20, No. 2, pp. 311-319, 1974.

[3] Rivas, J. R. e Rudd, D. F., Synthesis of Failure-Safe Operations, *AIChE Journal*, Vol. 20, No. 2, pp. 320-325, 1974.

[4] O'Shima, E., Safety Supervision of Valve Operations, *Journal of Chemical Engineering of Japan*, Vol. 11, No. 5, pp. 390-395, 1978.

[5] Foulkes, N. R., Walton, M. J., Andow, P. K. e Galluzzo M., Computer-Aided Synthesis of Complex Pump and Valve Operations, *Computer Chemical Engineering*, Vol. 12, No. 9/10, pp. 1035-1044.

[6] Tomita, S., Hwang, K. O'Shima, E. e McGreavy, C., Automatic Synthesizer of Operating Procedures for Chemical Plant by Use of Fragmentary Knowledge, *Journal od Chemical Engineering of Japan*, Vol. 22, No. 4, 364-372, 1989.

[7] Fikes, R. E. e Nilsson, N. J., Strips: A new Approach to the Application of Theorem Proving to Problem Solving, *Artificial Intelligence 2*, pp. 189-208, 1971.

[8] Sacerdoti, E. D., The Nonlinear Nature of Plans, em *Readings in Planning*, Eds. James Allen, James Hendler e Austin Tate, 1990.

[9] Stefik, M., Planning with constraints (MOLGEN: Part 1), em *Readings in Planning*, Eds. James Allen, James Hendler e Austin Tate, 1990.

[10] Fusillo, R. H. e Powers, G. J., A Synthesis Method for Chemical Plant Operating Procedures, *Computer Chemical Engineering*, Vol. 11, No. 4, pp. 369-382, 1987.

[11] Fusillo, R. H. e Powers, G. J., Operating Procedure Synthesis Using Local Models and Distributed Goals, *Computer Chemical Engineering*, Vol. 12, No.9/10, pp. 1023-1034, 1988.

[12] Lakshmanan, R. e Stephanopoulos, G., Synthesis of Operating Procedures for Complete Chemical Plants - I: Hierarchical, Structured Modelling for Nonlinear Planning, *Computer Chemical Engineering*, Vol. 12, No. 9/10, pp. 985-1002, 1988.

[13] Lakshmanan, R. e Stephanopoulos, G., Synthesis of Operating Procedures for Complete Chemical Plants - II: A Nonlinear Planning Methodology, *Computer Chemical Engineering*, Vol. 12, No. 9/10, pp. 1003-1021, 1988.

[14] Grefenstette, J. J., Genetic Algorithms and Their Applications, Em *The Encyclopedia of Computer Science and Technology*, A. Kent e J. G. Williams (Eds.), Vol. 21, pp. 139-152, 1990.

[15] Grefenstette, J. J., Optimization of Control Parameters for Genetic Algorithms, *IEEE Transacitons on Systems, Man and Cybernetics*, Vol. 16, No. 1, pp. 122-128, 1986.

[16] Grefenstette, J. J., Competition-Based Learning for Reactive Systems, *Proceedings of the DARPA, Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 348-353, Nov/1990, Morgan Kaufmann.

[17] Grefenstette, J. J., Incorporating Problem Specific Knowledge into Genetic Algorithms, Em *Genetic Algorithms and Simulated Annealing*, L Davis (ed.), pp. 49-67, 1987, Pitman Press.

[18] Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Company, 1989.

[19] Goldberg, D. E., Computer-Aided Pipeline Operation Using Genetic Algorithms and Rule Learning PART I: Genetic Algorithms in Pipeline Optimization, *Engineering with Computers*, No. 3, pp. 35-45, 1987.

[20] Goldberg, D. E., Computer-Aided Pipeline Operation Using Genetic Algorithms and Rule Learning PART II: Rule Learning Control of a Pipeline Under Normal and Abnormal Conditions, *Engineering with Computers*, No. 3, pp. 47-58, 1987.

[21] Goldberg, D. E., M. ASCE e Kuo, C. H., Genetic Algorithms in Pipeline Optimizations, *Journal of Computing in Civil Engineering*, Vol. 1, No. 2, pp. 128-141, 1987.

[22] Booker, L. B., Goldberg, D. E. e Holland, J. H., Classifier Systems and Genetic Algorithms, *Artificial Intelligence*, No. 40, pp. 235-282, 1989.

[23] Fujiki, C. e Dickinson, J., Using the Genetic Algorithm to Generate Lisp Source Code to Solve the Prisoner's Dilemma, *ICGA*, 1985, pp. 236-240.

[24] De Jong, K., On using Genetic Algorithms to Search Program Spaces, *ICGA*, pp. 210-216, 1987.

[25] Handley, S., The Genetic Planner: The Automatic Generation of Plans for a Mobile Robot Via Genetic Programming, capítulo 18 em *Genetic Programming II* de J. R. Koza, MIT, In press, 1994.

[26] Koza, J. R., Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.

[27] Koza, J. R., Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, 1994.

[28] Astrom, K. J., Anton, J. J. e Arzén, K. -E., Expert Control, *Automatica*, Vol. 22, No. 3, pp. 277-286, 1986.

[29] Astrom, K. J., Intelligent Control, *ECC 91 European Control Conference*, Grenoble, Jul/1991.

[30] Smuts, W. B. e MacLeod, I. M., Using Discrete AI Techniques for Designing a Real-World Control Supervisor, *IFAC - Artificial Intelligence in Real-Time Control*, Shenvang, pp. 31-35, 1989.

[31] Murata, T., Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE*, Vol. 77, No. 4, Abril/1989.

[32] Le Bail, J., Alla, H. e David R., Hybrid Petri Nets, *European Control Conference 91*, pp. 1472-1477, Grenoble, Jul/1991.

[33] David, R. e Alla, H., Petri Nets for Modeling of Dynamic Systems - A Survey, *Automatica*, Vol. 30, No. 2, pp. 175-202, 1994.

[34] David, R., Modelling of Dynamic Systems by Petri Nets, *European Control Conference 91*, pp. 136-147, Grenoble, Jul/1991.

[35] Maciel, P. R. M., Lins, R. D. e Cunha, P. R. F, Introdução às Redes de Petri e Aplicações, *X Escola de Computação*, Campinas, Jul/1996.

[36] Gentil, S., Systèmes d'aide à la Supervision In *Supervision de Processus à l'aide du Système Expert G2*^{TM}, Naly R. and Joseph A., Hermes Ed., pp. 7-20, Octobre/95, LAAS/CNRS.

[37] Homaifar, A., Qi, C. Q. and Lai, S. H., Constrained Optimization Via Genetic Algorithms, *Simulation*, pp. 242-254, April/1994.

[38] Fogel, D. B., A Comparison of Evolutionary Programming and Genetic Algorithms on Selected Constrained Optimization Problems, *Simulation*, pp. 397-404, June/1995.

[39] Park, S. K. e Miller, K. W., Random Number Generators: Goods Ones are Hard to Find, *Communications of ACM*, Vol. 31, N0. 10, pp. 1192-1201, Outubro/1988.

[40] Press, W. H., Flannery, B. P., Teukolsky S. A. e Vetterling, W. T., Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Capítulos 7, 15 e 16, 1988.