APLICAÇÃO DE METODOS DE INTELIGENCIA ARTIFICIAL AO PROBLEMA DE PLANEJAMENTO DE ROTAS

Ernesto Luis Villafuerte Oyola

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE POS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSARIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIENCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por :

Antonio de Almeida Pinho

Prof. Antonio de Almeida Pinho, D.Sc.
(Presidente)

Carlos Alberto da Silva Franco

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

Sheila Regina Morgel Veloso

Profa. Sheila Regina Morgel Veloso, D.Sc.

Claudia Q. R. do Valle

Profa. Claudia Guerreiro Ribeiro do Valle, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
JANEIRO DE 1989

VILLAFUERTE OYOLA, ERNESTO LUIS

Aplicação de Métodos de Inteligência Artificial ao Problema de Planejamento de Rotas [Rio de Janeiro] 1989 X, 280 p. 29,7 cm. (COPPE/UFRJ, M.Sc. Engenharia de Sistemas e Computação, 1989).

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Sistemas Baseados em Conhecimentos, Planejamento de Rotas I. COPPE/UFRJ II. Título (série).

A minha esposa Sissi e minha filha Suelen As quais muito sacrifiquei durante o desenvolvimento desta tese.

AGRADECIMENTOS

Ao Prof. Antonio de Almeida Pinho por sua grande ajuda, incentivo e orientação durante todo o desenvolvimento desta tese.

Ao Prof. João de Lizardo Araújo pelo incentivo e orientação na fase inicial de concepção da tese.

A Paulo César da Silva Seabra, Diretor do Departamento de Apoio Técnico (DPAT) do PRODERJ, que possibilitou a a realização deste trabalho.

Resumo da Tese Apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

APLICAÇÃO DE METODOS DE INTELIGENCIA ARTIFICIAL AO PROBLEMA DE PLANEJAMENTO DE ROTAS

Ernesto Luis Villafuerte Oyola

Janeiro/1989

Orientador : Antonio de Almeida Pinho

Programa : Engenharia de Sistemas e Computação

Neste trabalho é abordado o problema de planejamento de rotas em configurações com obstáculos. São apresentados e criticados, de forma resumida, os principais métodos de planejamento de rotas, existentes na literatura. Finalmente, é descrito um método que utiliza tecnicas de Inteligencia Artificial na modelagem do ambiente e do problema e na busca da solução. Também é apresentada uma implementação deste método na forma de Sistema de Produção, utilizando-se a linguagem LISP, descrevendo-se os principais algoritmos utilizados e analisando-se os resultados obtidos.

Abstract of the Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

APPLICATION OF ARTIFICIAL INTELLIGENCE METHODS FOR THE ROUTE PLANNING PROBLEM

Ernesto Luis Villafuerte Oyola

January/1989

Chairman : Antonio de Almeida Pinho

Department : System Engineering and Computation

The main methods for route planning with obstacles in up to date literature, are considered and analized.

A method for route planning is described. Artificial Intelligence techniques are used in this method to model environments, to describe the problem itself and the search for a solution. An implementation of such a method as a Production System using the LISP programming language, is pointed out. The main algorithms used in the implementation are presented as well as some tests to evaluate different aspects of the original problem.

INDICE

I- INTRODUÇAO 1

1.1 - Introdução 1

1.2 - Descrição do Problema 3

II- METODOS BASEADOS NA PROGRAMAÇAO MATEMATICA 7

2.1 - Algoritmo de Lee 8

2.2 - Variações do Algoritmo de Lee 14

2.3 - Planejamento Geométrico Detalhado 15

2.3.1 - Método de Lozano-Perez 16

2.3.2 - Alternativas para representação do grafo de busca 21

2.3.3 - Busca do caminho 26

2.4 - Análise dos Métodos Apresentados 27

2.4.1 - Considerações sobre os métodos de Lee e Hoel 27

2.4.2 - Considerações sobre os métodos que utilizam planejamento geométrico detalhado 30

III- REPRESENTAÇAO DO CONHECIMENTO : METODO DE YASUHIRO 39

3.1 - Representação do Espaço Livre 42

3.2 - Utilização do Conhecimento sobre Rotas 43

3.2.1 - Conhecimento diretamente aplicável na busca do caminho 46

3.2.2 - Conhecimento sobre solução de problemas 51

3.3 - Determinação da Rota 52

3.4 - Análise do Método de Planejamento de Rotas 52

IV- TOPICOS DE INTELIGENCIA ARTIFICIAL E O PROBLEMA DE PLANEJAMENTO DE ROTAS 56

4.1 - Resolução de Problemas 56

4.1.1 - Sistemas de Produção 60

4.1.2 - Estratégias de controle 61

4.1.3 - Decomposição do problema 64

4.1.4 - Busca heurística 65

4.2 - Representação do Conhecimento 67

4.2.1 - Lógica de predicados 68

4.2.1.1 - Unificação 69

4.2.1.2 - Resolução em lógica de predicados 69

4.2.1.3 - Lógica de predicados em I.A. 73

4.2.2 - Frames 77

4.2.3 - Representação do conhecimento e o "Frame Problem" 79

4.2.4 - Representação do conhecimento no problema de planejamento de rotas 80

4.3 - Linguagens de I.A. 83

4.3.1 - LISP 84

4.3.2 - OPS5 85

4.3.3 - PROLOG 85

V- MODULO DE INFERENCIA 88

5.1 - Aspectos Principais do Módulo de Inferência 88

5.2 - Codificação de Regras 94

5.3 - Interpretador de Regras 101

5.4 - Considerações Finais sobre o Módulo de Inferência 107

VI- REVISÃO DO METODO DE PLANEJAMENTO DE ROTAS 108

6.1 - Geração do Espaço Livre 108

6.2 - Descrição dos Principais Tipos de Restrições Aplicáveis na Busca do Caminho e outras Heurísticas sobre o Problema 111

6.3 - Função de Avaliação e Heurísticas 117

6.4 - Utilização do Conhecimento pelo Algoritmo de Busca do Caminho 123

6.5 - Relaxamento de Restrições 129

6.6 - Representação e Utilização do Conhecimento em outros problemas de Planejamento de Rotas 132

6.7 - Sistemas Especialistas 136

6.8 - Implementação do Sistema de Planejamento de Rotas 141

6.8.1 - Estruturas de Dados 146

6.8.2 - Algoritmo de Geração de Celas 151

6.8.3 - Algoritmo de Geração do slot "Restrições" do Frame da Rota 156

6.8.4 - Algoritmo de Determinação da Rota 157

6.8.5 - Adição de Novas Restrições, Funções de Avaliação e Atributos 159

VII- RESULTADOS EXPERIMENTAIS E CONCLUSÕES 166

7.1 - Resultados Experimentais 166

7.1.1 - Teste 1 175

7.1.2 - Teste 2 179

7.1.3 - Teste 3 183

7.1.4 - Teste 4 187

7.1.5 - Teste 5 191

7.1.6 - Teste 6 196

7.1.7 - Teste 7 200

7.2 - Análise dos Resultados Experimentais e Conclusões 207

REFERENCIAS BIBLIOGRAFICAS 214

APENDICE A - LISTAGEM DO CODIGO EM LISP DOS MODULOS DO SISTEMA DE PLANEJAMENTO DE ROTAS 218

CAPITULO I - INTRODUÇAO

I.1 INTRODUÇAO.

A elaboração de um plano ou estratégia de solução consiste em determinar uma sequência de ações a serem executadas com a finalidade de atingir determinado objetivo. No caso de problemas complexos decomponíveis, a solução implica em atingir previamente uma série de estados intermediários . A representação da solução global do problema pode ser feita através de um grafo de estados. E possível simular com o uso do computador a execução de uma série de ações, reavaliação destas ações e gerar outra sequência até atingir a solução. O jogo dos 8 é um exemplo de um caso no qual tal estratégia poderia ser aplicada. A construção de oleodutos por exemplo é uma tarefa onde a execução das etapas é irrevogável, tornando imprescindivel o planejamento prévio.

Os primeiros métodos de planejamento de rotas entre obstáculos, como os de LEE (1971) e HOEL (1976), utilizavam uma representação espacial bastante acurada embora ineficiente do ponto de vista computacional. Outros algoritmos que utilizam redes para modelar o problema, e buscam soluções ótimas, são tambem ineficientes. O uso de algoritmos baseados na programação heurística é característica de métodos posteriores como o de LOZANO-PEREZ (1979) e BROOKS (1985), conjuntamente com o uso de uma representação espacial mais eficiente, suficientemente precisa para a maioria dos casos ou cujas soluções podem servir de orientação para métodos mais precisos.

O método aqui apresentado é inspirado no

método de YASUHIRO (1986) que utiliza técnicas de inteligência artificial para representar o conhecimento sobre o problema, tais como critérios de layout e heurísticas. As estruturas usadas para representação do conhecimento serão descritas com detalhes assim como o mecanismo de inferência de novas informações.

Todo o capítulo V será dedicado à descrição do Motor de Inferência desenvolvido para o método e à sua implementação em LISP; foram realizadas algumas adaptações no método original para diminuir as possibilidades de fracaso na determinação de rotas e melhorar sua "pericia" na determinação das melhores soluções possíveis, relaxando restrições quando for conveniente. A estrutura usada para representar os índices de performance proporciona ao método a capacidade real de admitir um número grande e variado destes índices, assim como a de utilizar a combinação desejada destes.

Neste método, a busca da solução gera um grafo de estados no qual cada nó representa um estado diferente na configuração e onde a passagem de um estado para outro é "decidida" pelo método com base no conhecimento disponível nas estruturas de representação do conhecimento. Estas estruturas serão examinadas no capítulo IV. Finalmente, o método utiliza conhecimento em dois níveis : conhecimento utilizavel na busca do caminho e conhecimento sobre o próprio processo de solução do problema.

A tese é organizada em 7 capítulos, descritos a seguir: o capítulo I é dedicado a descrição do problema abordado. No capítulo II são apresentados de forma resumida, os principais métodos até agora desenvolvidos para a

determinação de rotas em configurações com obstáculos. Esta apresentação servirá para ilustrar os diferentes tipos de abordagens que são utilizadas. No capítulo III é descrito o método original de planejamento de rotas que utiliza representação do conhecimento e que possibilita uma representação mais completa sobre as características da configuração e do conhecimento que o projetista possui sobre o problema.

No capítulo IV são tecidas algumas considerações sobre o próprio processo de resolução de problemas, estruturas de representação do conhecimento e justificativa de sua utilização no método de planejamento de rotas. No capítulo V descreve-se as técnicas utilizadas no desenvolvimento do Módulo de Inferência usado pelo método de planejamento de rotas e dos principais algoritmos utilizados na sua implementação. No capítulo VI são apresentadas adaptações de alguns aspectos do método de planejamento de rotas e é considerada a possibilidade de adaptação de outros métodos para o uso de técnicas de I.A.. Por último, no capítulo VII são apresentados e analisados os resultados experimentais obtidos, assim como as conclusões finais da tese.

I.2 DESCRIÇÃO DO PROBLEMA.

A maior parte dos métodos de busca do caminho entre obstáculos existentes na literatura foram desenvolvidos visando áreas de aplicação específicas. Neste trabalho visamos principalmente aplicações em áreas tais como planejamento de rotas para tubulações, caminhos para um robô móvel, projeto de conexões LSI e no projeto de layout

de plantas industriais. O planejamento de rotas para tubulações envolve, por exemplo, considerações sobre algumas propiedades dos fluidos a serem escoados; ou no caso do caminho para um robô móvel, sobre propriedades do robô ou da carga que é transportada por este. Para efeito de simplificação , estas propriedades são consideradas propiedades da rota.

As abordâgens baseadas nos algoritmos da programação matemática possuem aplicação limitada a casos em que são utilizados indices de performance e restrições uniformes e muito simples. Problemas do mundo real porêm são muito mais complexos pois envolvem flexibilidade na escolha de indices de performance e restrições, uma estratêgia global que incorpore o conhecimento sobre o problema e o ambiente; este conhecimento envolve os critêrios de layout e heuristicas que o projetista utiliza no planejamento de rotas.

No projeto de layout de plantas industriais são utilizados dois mêtodos altamente conversacionais; no primeiro, os projetistas interagem com um modelo reduzido em escala da planta para determinar a rota ôtima de acordo com suas necessidades. O outro mêtodo utiliza computação grâfica. Através de uma "estação grâfica", que dispõe de uma mesa digitalizadora, teclado e telas de video, o projetista interage com um modelo geomêtrico computadorizado mostrado em um terminal CRT. Um equipamento com duas telas, cada uma divisivel em quatro, permite trabalhar em oito detalhes ou vistas de um mesmo desenho; este mêtodo pode ser mais eficiente, desde que o modelo é computadorizado, e pode ser mais automâtico no processo de recuperação de dados. Poucos

sistemas deste tipo porêm, oferecem alguma função de suporte que permita incorporação do conhecimento sobre o processo de planejamento de rotas. Este deveria ser automatizado para reduzir a quantidade de iterações homem-maquina e melhorar a produtividade.

A elaboração de um método eficiente de planejamento de rotas para a solução dos problemas nas áreas mencionadas passa pela geração de uma boa representação espacial do ambiente do problema. A representação espacial gerada deve ser o mais econômica possivel, para evitar um consumo elevado de memória, mas possuir um nivel de precisão suficiente para resolver a grande maioria dos problemas formulados.

A necessidade de representar o conhecimento do projetista sobre o problema, nos conduz à utilização de técnicas de inteligência artificial de representação do conhecimento. As estruturas utilizadas devem permitir o armazenamento e recuperação eficiente da informação quando necessário, e o método a ser utilizado deve permitir a incorporação de um número grande e variado de restrições, índices de performance e heurísticas sobre o problema; ao mesmo tempo, o método deve permitir a seleção flexivel destes índices e restrições.

Os processos de determinação de caminho em redes podem acarretar tempos elevados de processamento, devido à natureza combinatória envolvida. Por esse motivo deve ser usado, na determinação da rota, um algoritmo eficiente, que permita a incorporação de heurísticas sobre o problema, com o objetivo de melhorar o desempenho computacional. O próprio processo de resolução de problemas

deve ser estudado para possibilitar a incorporação de estratégias que ajudem a melhorar a eficiência do método.

O Método utilizado neste trabalho será descrito para o caso bidimensional (Coordenadas X-Y). O layout possui limites retangulares e os objetos da configuração possuem traços retos e arestas alinhadas com os eixos coordenados.

O objetivo do problema é achar uma rota Manhattan viável entre dois blocos da configuração. Isto é, apenas 4 direções para a rota são possíveis (Norte, Leste, Sul, Oeste). A extensão do método para casos mais abrangentes como a possibilidade de seguir 8 direções para o caso bi-dimensional ou 26 para o tridimensional será abordada superficialmente mas não se encontra no escopo deste trabalho.

O capítulo seguinte permitirá uma visão dos recursos e limitações das abordagens constantes na literatura, que foram utilizadas na solução de problemas simples de busca da rota ótima em um espaço bi-dimensional com obstáculos.

CAPITULO II - METODOS BASEADOS NA PROGRAMAÇAO MATEMATICA

Neste capítulo é feita uma breve revisão dos principais métodos existentes na literatura para resolução do problema de busca de caminho entre obstáculos. Os métodos aqui abordados utilizam estratégias de solução baseadas em algoritmos da programação matemática.

A fim de proporcionar uma visão global da evolução das estratégias utilizadas, estes métodos são aqui apresentados em ordem cronológica do seu surgimento na literatura.

O método de LEE (1971) é o primeiro na cronologia dos métodos aqui apresentados. HOEL (1976) sugeriu algumas variações do método de Lee, especificamente, do algoritmo de busca do caminho e das estruturas de dados por este algoritmo utilizadas, embora a representação spacial da configuração problema seja essencialmente a mesma que a utilizada por Lee.

Os métodos de LOZANO-PEREZ (1979) e BROOKS (1983) são mais recentes e trazem como principais inovações, uma representação espacial mais eficiente e a utilização de algoritmos de busca que permitem a incorporação de funções heurísticas de avaliação. O tratamento de deslocamento de objetos, articulados ou não, sujeitos ou não a rotações é possível com a utilização destes métodos e tem a sua aplicação principal em robótica. Continua sendo porém uma característica comum dos métodos apresentados neste capítulo, a rigidez do processo de resolução de problemas inerente aos métodos da programação matemática.

2.1 Algoritmo de Lee.

LEE (1971) desenvolveu um dos primeiros algoritmos para a solução do problema de caminho mínimo com obstáculos, visando especificamente aplicações em diagramação elétrica e desenhos lógicos. O tipo de obstáculo considerado pelo algoritmo são arestas alinhadas com os eixos coordenados ou obstáculos retângulares com arestas também alinhadas com os eixos coordenados. Este algoritmo foi até pouco tempo atrás a base da maioria dos algoritmos utilizados na área de projeto lógico de circuitos eletrônicos.

O algoritmo descrito nesta seção tem sua aplicação numa representação espacial da configuração formado por celas planas quadradas homogêneas representadas por meio de uma matriz bi-dimensional. Através de um mapeamento é atribuida a cada cela um "conteúdo" que pode ser um elemento de um conjunto de símbolos definidos previamente (p. ex.: letras do alfabeto, números de 0 a 9, os símbolos : !,-,!¯,¯!,!_,_!,X,->,<-,branco,etc.), que servem para identificar, entre outras coisas, se uma cela encontra-se disponível, constitui cela proibida, ocupada (caso faça parte da aresta de um obstáculo) ou se ja foi atingida durante a busca do caminho e a partir de que direção foi atingida. E possível utilizar este mapeamento para formar um grupo de celas admissíveis e outro de celas não admissíveis de acordo com as restrições do problema. A figura II.1 mostra o mapeamento inicial para uma configuração exemplo antes de aplicar o algoritmo de busca. A convenção utilizada neste exemplo é a seguinte :

FIGURE 11.1

96 x	88 x	80 x	72 x	64 x	56 x	48 x	40 x
95 x	87 x	79 x	71 x	63 x	55 x	47 x	39 x
94 x	86 x	78 x	70 x	62 x	54 x	46 x	38 x
93 x	85 x	77 x	69 x	61 x	53 x	45 x	37 x
92 x	84 x	76 x	68 x	60 x	52 x	44 x	36 x
91 x	83 x	75 x	67 x	59 x	51 x	43 x	35 x
90 x	82 x	74 x	66 x	58 x	50 x	42 x	34 x
89 x	81 x	73 x	65 x	57 x	49 x	41 x	33 x

|,-,|¯,¯|,|_,_| => vértices e parte das arestas de um obstáculo.

X => cela proibida (não admissível)

branco => cela vazia disponível

A definição dos principais termos utilizados na descrição do algoritmo de Lee é dada a seguir :

Lista de celas : Uma lista de celas L é uma lista ordenada contendo nomes de celas.

Massa de celas : Com cada cela da lista L será associada um vetor de dimensão r (r é um inteiro positivo correspondente ao número de funções de avaliação existentes para o problema), chamado massa da cela $(m_1, m_2, \ldots, m_r)$, obtida pela aplicação das funções de avaliação $(f_1, f_2, \ldots, f_r)$. As massas das celas estão ordenadas lexicograficamente. Assim, se $m = (m_1, m_2, \ldots, m_r)$ e $m' = (m_1', m_2', \ldots, m_r')$ são duas massas de celas, $m < m'$ se $m_i = m_i'$, $i = 1, 2, \ldots, k$; mas $m_{k+1} < m'_{k+1}$ para k pertencente ao intervalo $[1, r[$.

Cela vizinha : Para uma cela qualquer c_i da configuração existem 4 celas vizinhas possíveis, uma para cada direção admitida pelo algoritmo de busca do caminho : norte, oeste, sul e leste.

Cadeia : Denominamos de cadeia ao conjunto de celas que formam o caminho $p(c^i, c^j) = (c^o=c^i, c^1, \ldots, c^m=c^j)$ tal que c^{i+1} pertence aos vizinhos de c^i para $i = 0,1,\ldots,m-1$.

Coordenada de cadeia : Com cada cela da lista L está também associado um símbolo de coordenada de cadeia pertencente ao conjunto de símbolos definidos para o problema e que serve para identificar a direção a partir da qual uma determinada cela foi atingida durante o processo de busca do caminho.

Lista auxiliar L_1 : Lista auxiliar utilizada para armazenamento temporário de nomes de celas.

O algoritmo é descrito em um nível abstrato em que são possíveis de aplicação r funções de avaliação, para um r inteiro positivo qualquer. Um caminho é dito admissível se satisfaz as restrições do problema. Os passos do algoritmo são os seguintes :

a) A cela inicial c^* é dada a massa $(0,0,\ldots,0)$. A lista L contém inicialmente o único membro c^*. A lista L_1 está inicialmente vazia.

b) Enquanto a cela objetivo c^{**} não aparece na lista L e a lista L não está vazia faça :

b.1) Seja c uma cela na lista L. Por um caminho

$p(c^*,c^1,c^{**})$ nós queremos-nos referir a um caminho $p(c^*,c^{**})$ do qual $p(c^*,c^1)$ é um sub-caminho. Uma cela c^1 é dita admissível se $p(c^*,c^1,c^{**})$ é um caminho admissível e se a massa da cela para c^1 não foi ainda determinada.

Seja $\{c^1\}$ pertencente ao conjunto de vizinhos da cela c, o conjunto de todas as celas admissíveis. Adicione à lista L_1 o conjunto $\{c^1\}$. L_1 é construida pela repetição desta operação de adição para cada cela c da lista L, sob a condição que uma cela não deve ser listada mais de uma vez. L_1 é portanto o conjunto de todas as diferentes celas c^1 tal que c^1 é uma cela admissível pertencente ao conjunto de vizinhos de c, para alguma cela c na lista L.

b.2 (Procedimento para atribuir massas às celas e coordenadas de cadeia)

Seja c^1 uma cela em L_1. Uma possível massa da cela para c^1 é determinada como segue :

Para cada c^j que pertence ao conjunto de vizinhos de c^1 cuja massa de cela tenha sido determinada construa um vetor de dimensão igual ao número r de funções de avaliação f existentes para o problema,

$$(f_1(p(c^*,c^j,c^1)),\ldots f_r(p(c^*,c^j,c^1)))$$

Agora apliquemos a regra de seleção b.3 para achar um c^{j^o} que pertence ao conjunto de vizinhos de c^1 para o qual esta r-tupla é mínima.

Uma possível massa de cela para c^1 é então a r-tupla :

$$(f_1(p(c^*,c^{j^o},c^1)),\ldots f_r(p(c^*,c^{j^o},c^1))),$$

e uma possível coordenada de cadeia para c^i é uma seta pertencente ao conjunto de símbolos definidos inicialmente para o problema que indica a direção a partir da qual a cela c^i foi atingida partindo da cela $c^{j<i}$. A seguir, achar todas as possíveis massas de cela para todo c^i que pertence à L_1.

Entre as celas em L_1, seja $\{c^L\}$ o conjunto das celas cujas possíveis massas de celas são mínimas. Nós atribuimos para todo c^L que pertence à $\{c^L\}$ a mesma massa de cela e a mesma coordenada de cadeia como sua possível coordenada de cadeia. As celas cujas possíveis massas de cela não são mínimas não são mais consideradas e a lista L_1 é "limpada".

Sejam por exemplo os seguintes vetores, as r-tuplas obtidas para cada um dos vizinhos admissíveis de c^i para r = 5 :

	f_1	f_2	f_3	f_4	f_5
c^{i+1} ==>	2	2	1	0	0
c^{i+2} ==>	2	1	0	0	0
c^{i+3} ==>	2	1	1	0	0

Como as massas das celas são ordenadas lexicográficamente a ordem das massas é a seguinte :

$$m(c^{i+2}) < m(c^{i+3}) < m(c^{i+1})$$

e consequentemente $m(c^{i+2})$ é a r-tupla mínima. Observe que embora $m(c^{i+2})$ e $m(c^{i+3})$ possuam os valores de f_1 e f_2 idênticos, o valor de f_3 de $m(c^{i+2})$ é menor que o valor de f_3 de $m(c^{i+3})$ e portanto $m(c^{i+2}) < m(c^{i+3})$.

b.3 Em b.2, uma cela c^i e um sub-conjunto $\{c^j\}$ do

conjunto dos vizinhos de c^4 foi dado. A regra que foi invocada para selecionar um c^{j0} apartir do conjunto $\{c^j\}$ é chamada de regra de seleção (Esta regra varia de acordo com o tipo de problema abordado).

b.4 (Atualização da lista L).

Seja $\{c^k\}$ o conjunto de celas cujas massas foram determinadas em b.2. Concatene primeiro à lista L o conjunto $\{c^k\}$. A seguir examine cada cela c em L para verificar se as massas de todos os vizinhos desta cela foram determinadas. Caso afirmativo, apague c da lista L.

2.2 Variações do algoritmo de Lee.

Durante a busca do caminho pelo algoritmo de Lee, é gerada o que Lee denomina de "onda de busca". As celas que se encontram no limite de tal "onda" são chamadas de celas fronteiriças e que equivalem aos nós folhas de uma árvore de busca. HOEL (1976) mostra que armazenando as celas fronteiriças em um "array" de pilhas ao invés de uma única lista, operações de busca de celas, de custo computacional elevado, podem ser eliminadas com um aumento significativo da demanda de capacidade de armazenamento. Ele mostra também que se o custo do caminho é a soma dos "pesos" de suas celas, ponteiros que indicam o antecessor de uma cela podem ser atribuidos as celas assim que estas são atingidas em vez de quando são expandidas. O "peso" de uma cela é o resultado da aplicação nesta cela de uma função de avaliação W. Esta função W pode, no caso da função de avaliação ser a métrica do caminho, computar o incremento na métrica do caminho

ocasionado pela seleção de uma cela c qualquer como segmento do caminho.

O algoritmo sugerido por Hoel que incorpora estas variações foi denominado por ele de algoritmo X. O algoritmo X seleciona as celas fronteiriças para expansão na ordem crescente do custo do caminho porque uma cela nunca precisa ser expandida mais de uma vez; o custo de uma cela pertencente ao caminho determinado pelo algoritmo X é conhecido quando a cela é expandida, e o custo do caminho não precisa ser armazenado. a ideia de expandir celas fronteiriças nesta ordem é geralmente creditada a Dijkstra.

As celas que compõem a representação espacial da configuração problema, são concebidas da mesma forma que no método de LEE (1971), como celas quadradas homogêneas e representadas computacionalmente através de uma matriz bi-dimensional.

Outras variações do algoritmo de Lee são sugeridas para contextos especiais. Alternativas para melhorar a eficiência da busca, utilizando estruturas de dados tais como pilhas e vetores são também sugeridas por Hoel, para aumentar a eficiência da localização de celas fronteiriças e consequêntemente aumentar tambem a eficiência do processo global de determinação do caminho.

2.3 - Planejamento Geométrico detalhado.

Nas seções anteriores vimos o algoritmo de Lee que resolvia o problema genérico de caminho livre de colisões a custo de grande esforço computacional e consumo de memória. A abordagem de Lee considerava o estabelecimento de ligações de um ponto a outro qualquer mas não considerava

a possibilidade de planejar estas mesmas ligações quando havia movimentação de um objeto. (Assumia-se, quando fosse o caso, que o objeto movimentado era adimensional, ou seja, um ponto). É necessário a introdução deste aspecto nas nossas considerações sobre busca do caminho posto que muitos dos problemas do mundo real envolvem deslocamento de objetos cujas dimensões não podem ser desprezadas. A solução deste tipo de problema inclui o tratamento de duas ou mais dimensões, que envolvem p. ex. deslocamento nos eixos coordenados, movimento de rotação e/ou translação. LOZANO-PEREZ (1979), BOYSE (1979), BROOKS (1983) e outros utilizaram este tipo de abordagem. É tipico destes métodos tentar modelar o ambiente do problema com níveis de precisão detalhados. Como objeto de análise desta classe de métodos, será apresentado um dos métodos usados por LOZANO-PEREZ (1979 e 1981), fazendo-se ulteriormente uma crítica do método procurando os aspectos comuns com outros métodos que utilizam este tipo de abordagem.

2.3.1 Método de Lozano-Perez.

Este método tem por objetivo resolver o problema de busca do caminho entre obstáculos poliédricos convexos, onde o objeto movimentado também é poliédrico convexo. Lozano-Perez utiliza o algoritmo A* para percorrer o espaço de busca assim definido.

A razão da modelagem do ambiente da configuração através de objetos poliédricos convexos provavelmente reside na tentativa de usar um modelo do mundo real mais preciso que o tradicional "mundo de blocos", uma vez que a utilização de objetos poliédricos permite uma

representação mais precisa de uma configuração no mundo real.

O método usado tem aplicabilidade geral, o que faz com que seja mencionado em nosso trabalho embora Lozano-Perez tenha pensado posteriormente em aplicações mais específicas, como Robótica. Uma abordagem mais específica dos problemas de robótica é feito em LOZANO-PEREZ (1979, 1981 e 1982).

Uma maneira de solucionar o problema de busca de caminhos para um objeto móvel seria computar explicitamente as restrições na posição do objeto sendo movimentado, em relação aos obstáculos; a trajetória desejada seria então o caminho mais curto que satisfaz todas as restrições de posição. Se os objetos são modelados como coleções de poliedros convexos, as restrições de posição podem ser estabelecidas em termos da posição dos vértices do objeto movimentado relativamente aos planos das superfícies do obstáculo. O problema da trajetória poderia então ser modelado como um problema de otimização, como em IGNAT'YEV (1973). A dificuldade existente com esta formulação é que estas restrições de posição, apesar de lineares, não se aplicam simultâneamente. Assim não é necessário que cada ponto no objeto movimentado esteja fora de todos os planos dos obstáculos; é suficiente que cada ponto esteja fora de pelo menos um dos planos de cada obstáculo. Esta propriedade torna inaplicáveis os métodos tradicionais de otimização linear. Apesar disto, o método de Lozano-Perez está proximo da abordágem otimizacional. Feitas estas considerações podemos dar a definição formal do problema : Seja R um poliedro convexo que limita o espaço de trabalho onde

existem outros K_b obstáculos B_j, possivelmente sobrepostos denominados obstáculos e seja A o objeto sendo movimentado, a união de K_a poliedros convexos A_i, i.e.:

$$A = \bigcup_{i=1}^{K_a} A_i \ .$$

O problema da determinação de um caminho em uma configuração com obstáculos pode então ser sub-divido em duas etapas :

1) Determinação do espaço livre.- Achar uma configuração para A, interna a R, tal que para todo i e para todo j : A_i interseção $B_j = 0$. Esta é denominada uma configuração segura.

2) Busca do caminho.- Achar um caminho para A da configuração s para a configuração g tal que A esteja sempre em R e todas as configurações de A no caminho sejam seguras. Este é denominado um caminho seguro.

O número de parâmetros requeridos para especificar a configuração de um poliedro k-dimensional, A, relativamente à sua configuração inicial é d, onde :

$$d = k + C(k,2)$$ [LOZANO-PEREZ (1983)]

C(k,2) -> número de combinações de k, 2 a 2 e igual ao número de ângulos necessários para especificar a rotação relativa de um sistema de coordenadas em relação a outro. Este número tambem é denominado de ANGULO DE EULER.

A configuração para um sólido tridimensional pode então ser representada através de 6 coordenadas.

$$d = 3 + C(3,2) = 6$$

O espaço de busca hexa-dimensional de configurações para um sólido A é denominado ESPAÇO DA CONFIGURAÇÃO e é denotado $Cespaço_A$. Uma configuração pode ter, por exemplo, para o caso do sólido tridimensional, um valor de coordenada para cada uma das coordenadas x, y, z de um ponto selecionado no objeto (este ponto pode ser um vértice de referência como será visto mais adiante) e um valor de coordenada para cada um dos ângulos de Euler.

No $Cespaço_A$, o conjunto de configurações de A onde A se sobrepõe a B, i.e., A interseção B diferente de nulo, será denotado $CO_A(B)$, o obstáculo do $Cespaço_A$ devido a B. Analogamente, aquelas configurações de A onde A está completamente contido em B, serão denotadas $CI_A(B)$, o $Cespaço_A$ interior de B. Em duas dimensões, se a orientação de um poligono convexo A é fixada, o $Cespaço_A$ é simplesmente o plano (x,y), isto porque a posição (x,y) de algum vértice de referência, rv_A, é suficiente para especificar a configuração do poligono. Neste caso, a presença de outro poligono convexo B restringe rv_A a ficar fora de $CO_A(B)$, um poligono convexo maior mostrado como a região sombreada da fig. II.2. Como $CO_A(B)$ neste caso é um conjunto de valores (x,y), é denotado $CO_A^{xy}(B)$. Para multiplos obstáculos B_j, a localização de A é segura se e somente se rv_A não se encontra contido em $CO_A^{xy}(B)$, mas é interno à $CI_A^{xy}(R)$.

FIG. 12

O método sugerido por LOZANO-PEREZ (1979) determina o espaço livre realizando uma operação de "crescimento" de obstáculos cujo objetivo é computar explicitamente as regiões proibidas. Para o caso de um objeto poligonal móvel A de orientação fixa e multiplos obstáculos B_j, são computados explicitamente os $CO_A^{xy}(Bj)$. O algoritmo de crescimento mais simples move cada um dos lados do obstáculo original por uma quantidade constante r_A (que no caso de um objeto A circular trata-se do seu raio r) e então intersecta os novos lados obtidos por esta operação de deslocamento, para obter os vértices do poligono "estendido" (fig. II.3). Este método, usado por UDUPA (1977), foi modificado por Lozano-Perez para tratar do caso em que os objetos da configuração são poligonos convexos.

Em robótica, cada movimento distinto do braço de um robô é chamado de "grau de liberdade". Lozano-Perez estendeu este método para problemas com até sete graus de liberdade.

Nesta apresentação do método, é considerado o objeto movimentado como sendo do tipo simples, não-articulado, sujeito a movimentos de translação. Este é um caso que pode ser tratado de forma mais genérica, o que será de utilidade para a visualização da aplicaçao do método para o problema de busca do caminho de uma forma geral. Quando as dimensões do objeto movimentado são irrelevantes, este pode ser considerado como um ponto e a operação de crescimento de obstáculos se torna desnecessária.

2.3.2 Alternativas para representação do grafo de busca.

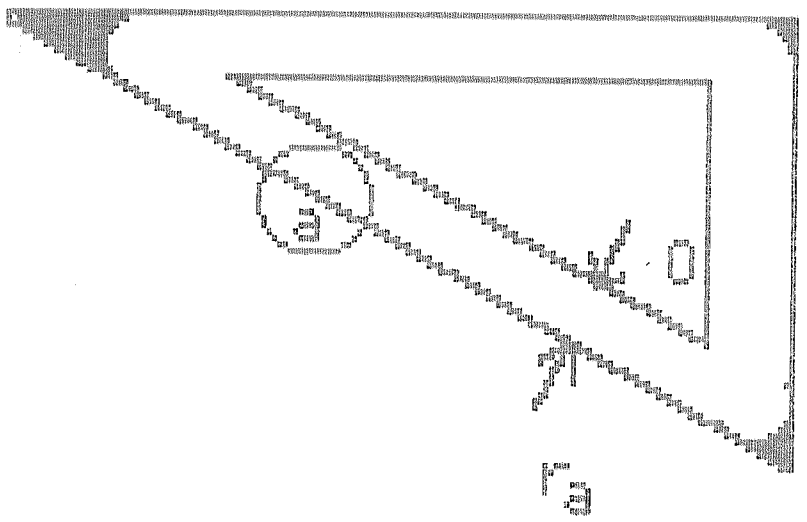
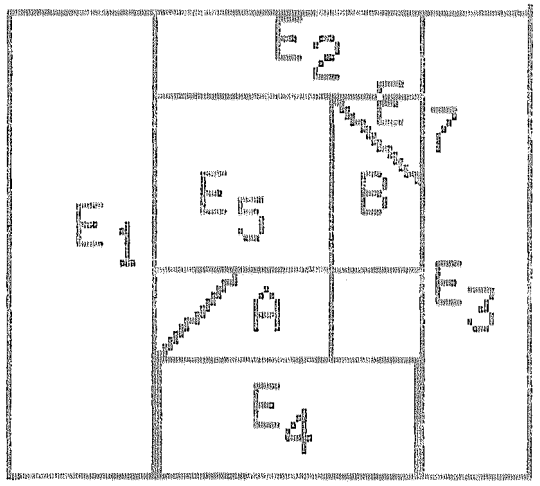


FIG. 11.3

Lozano-Perez sugere duas alternativas para representação do grafo de busca. Inicialmente é sugerido o uso do VGRAPH. O VGRAPH ("visibility-graph") tem como conjunto de vértices os pontos formados pelas posições de rV_A nas configurações inicial e final assim como os vértices dos obstáculos redefinidos $CO_A^{XY}(B_j)$. Os arcos são formados ligando todos os vértices no grafo com a condição de que a linha reta traçada de cada vértice i para j não intersecte nenhum $CO_A^{XY}(B_j)$. No caso de três dimensões, o caminho mais curto entre obstáculos poliédricos em geral não percorre somente vértices do poliedro. Em geral, o caminho mais curto livre de colisões implica em passar por pontos intermediários nas arestas dos obstáculos. A sugestão dada em LOZANO-PEREZ (1983) é a de adicionar vértices ao longo das arestas dos $CO_A^{XYZ}(B_j)$, com a finalidade de obter uma boa aproximação da solução ótima.

Uma desvantâgem da abordagem baseada no uso do VGRAPH no processo de busca é sua dificuldade na incorporação de heurísticas sobre o problema. Uma outra alternativa é a utilização do complemento dos obstáculos, chamado de espaço livre. Esta abordagem facilita o uso de variadas heurísticas de busca . A representação usada é uma representação hibrida que emprega dois tipos de celas : celas sólidas retângulares alinhadas com os eixos e poliedros convexos arbitrários. As celas retângulares simples serão usadas quando se estiver longe dos obstáculos, onde a economia de representação é importante; quando se estiver perto do obstáculo onde é necessária grande precisão, serão usadas celas poliédricas. E gerada uma árvore de distribuição de celas (figs. II.4a e II.4b). A



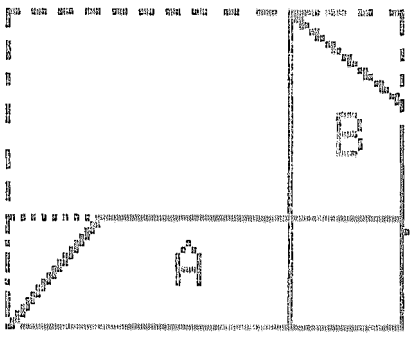


FIG. 11.4a

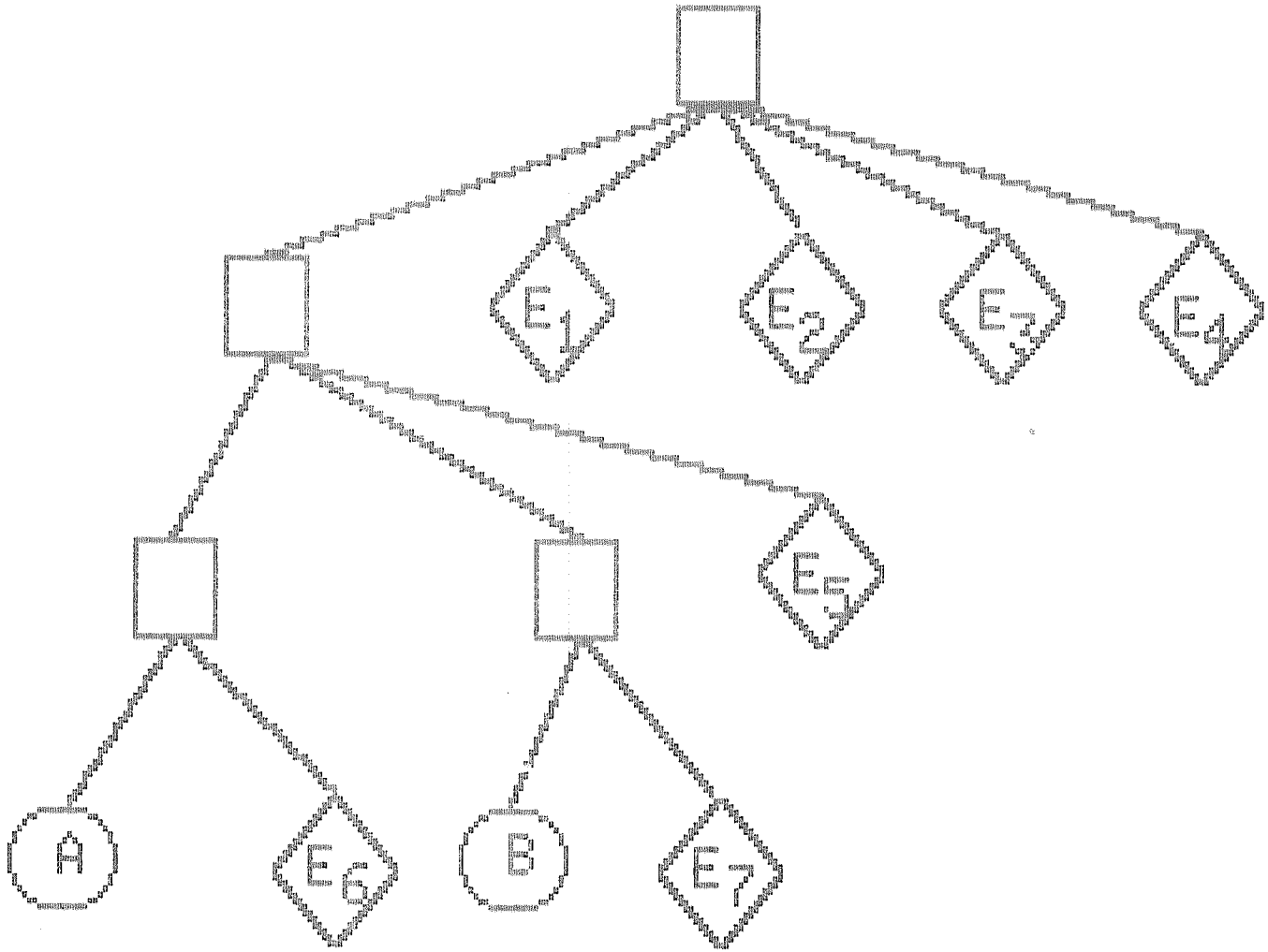


FIG. 11.4b

representação da distribuição da árvore usa celas retângulares como nós internos e celas poliédricas como folhas. As folhas representam o espaço que está cheio, i.é., completamente ocupado por um objeto. As celas internas representam espaço do tipo misto, i.é.,celas que são em parte cheias e em parte vazias.

A representação espacial é construida partindo de um sólido retângular limitado representando o espaço de trabalho; esta é a primeira cela mista. Os descendentes deste nó são as celas mistas correspondendo às raizes das arvores representando cada um dos $CO_A^{XYZ}(B_j)$, e um conjunto de sólidos retângulares vazios que fazem limite com as celas mistas representando o espaço livre em torno destas. A representação de cada cela mista pode ser posteriormente expandida em outras celas vazias, mista e cheias, culminando em uma representação envolvendo somente celas poliédricas convexas do tipo vazia ou cheia como nós-folha e celas do tipo mista como nós internos. A representação poliédrica de cada cela vazia deve ser computada de forma tal que esta não se sobreponha a nenhuma cela do tipo cheia ou mista.

Gerada a representação de celas vazias podemos construir o grafo FSG (Free Space Graph) formado pelos nós representando celas vazias, e estabelecendo arcos entre as celas quando estas se limitam entre si.

2.3.3 Busca do Caminho.

A busca do caminho com base na representação gerada é feita mediante as seguintes etapas :

a) Escolha a maior cela vazia da representação espacial

gerada inicialmente na seção anterior. Se não há disponivel nenhuma, escolha uma cela do tipo mista contendo a configuração inicial e decomponha-a nos seus componentes (celas vazia, mista ou cheia). Se uma cela vazia contêm a configuração inicial, pare. Senão repita o processo até achar uma. Se nenhuma cela do tipo vazia é achada, a tarefa é impossivel pois a configuração inicial causa uma colisão.

b) Execute o passo a) para a configuração final (objetivo).

c) Construa o grafo de espaço livre como descrito na seção anterior.

d) Faça a busca do caminho utilizando algum algoritmo convencional de busca em grafos ou o A*.

e) Escolha um traçado contido no caminho de celas obtido.

Para recuperar o caminho obtido em e) pode-se ligar os centroides de celas vizinhas neste caminho. Como esta técnica pode implicar em que a linha que une os centroides passe por fora do caminho como na fig. II.5a, uma alternativa é fazer as ligações entre os centroides das interseções de duas celas vizinhas (fig. II.5b).

2.4 Análise dos métodos apresentados.

2.4.1 Considerações sobre os Métodos de Lee e Hoel.

A exemplo do método de Lee e de outros métodos que utilizam processos tradicionais, a estrutura usada para representar a configuração espacial gera um número muito grande de celas até alcançar um nível de detalhamento suficiente para não perder informaçao. No caso

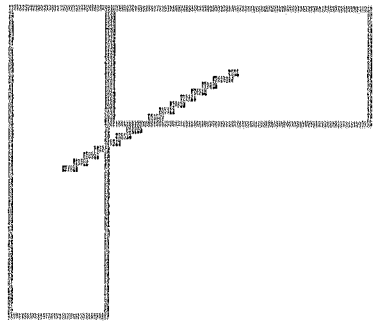


FIG. 11.5a

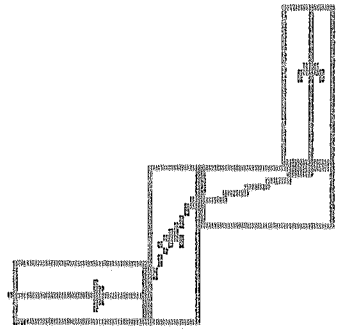


FIG. 11.5b

dos algoritmos de Lee e Hoel, a representação é feita por meio de celas quadradas homogêneas. O número de celas gerado é muito grande o que o torna ineficiente desde o ponto de vista da representação; especialmente para problemas onde o número de componentes da configuração é grande. Neste caso o custo computacional da busca utilizando esta representação pode tornar-se proibitivo. No caso específico do método de Lee, existem outros aspectos que o tornam também ineficiente desde o ponto de vista da busca realizada. No passo b.2 do algoritmo de Lee percebemos que as entradas das celas que não pertencem ao conjunto (c^1) de massas mínimas são apagadas na lista L_1. Isto ocasiona com que no passo b.1, uma mesma cela possa ser gerada mais de uma vez por uma mesma cela "pai" durante a aplicação do algoritmo. Conseqüentemente a determinação da massa de uma cela para esses casos, tambem é feita mais de uma vez. Esta característica tende a piorar o desempenho do algoritmo em relação ao caso médio verificado para outros algoritmos de complexidade da mesma ordem (exponencial). O método definido desta maneira é semelhante a uma busca em largura ("breath-first"). Não é por acaso que o próprio Lee faz uma analogia da expansão das celas com o de uma onda em expansão e a chama de "onda de busca". O tipo de funções de determinação de massa usados por Lee procuram a otimalidade de uma solução sacrificando em eficiência o método, que poderia ser melhorado, como será visto mais tarde, pelo uso de funções menos precisas, mas que garantiriam a obtenção de uma boa solução na maioria dos casos. Esta abordagem, que procura a otimalidade da solução, é característica dos métodos de programação matemática.

Em relação às modificações introduzidas por Hoel, podemos dizer que as restrições referentes à representação gerada e à característica da busca de otimalidade permanecem, embora alguns aspectos do algoritmo semelhante ao de programação dinâmica de Dijkstra, o torne mais eficiente que o algoritmo de Lee. Por outro lado, a introdução de índices de performance variados para cada caso particular sugerida por Hoel, implica necessàriamente em uma reavaliação cuidadosa do método utilizado.

O momento em que as desvantàgens mencionadas começam a ficar palpàveis é quando o número de elementos do problema cresce em tamanho e complexidade. A maioria dos problemas reais envolve um número grande e variado de elementos. As características dos problemas reais serão abordadas com mais detalhe posteriormente.

2.4.2 Considerações sobre os métodos que utilizam planejamento geométrico detalhado.

Ao invés de apresentar uma série de métodos que pertencem a esta classe, nos limitamos a apresentar um dos métodos utilizados por Lozano-Perez para o caso em que o objeto não está sujeito a rotações. É mais produtivo que apresentar cada um destes métodos, tentar estabelecer quais os principais pontos em comum entre eles.

A razão da escolha do método de Lozano-Perez é a de que este representa um dos principais métodos utilizados para resolver problemas de busca do caminho com obstáculos, principalmente na área de robótica. Este método

representa uma evolução em relação a métodos como o de BOYSE (1979) que utilizam correções locais quando há deteção de obstáculos, abordagens do tipo "generate-and-test" como a de TAYLOR (1979), e abordagens que utilizam métodos baseados em otimização linear como IGNAT'YEV (1973).

Os métodos que utilizam planejamento geométrico pretendem representar, ou implicitamente como em LOZANO-PEREZ (1979) ou explicitamente como em BROOKS (1983), o espaço livre de obstáculos da configuração espacial. O método de Lozano-Perez que foi apresentado neste capitulo calculava explicitamente as restrições nas posições do objeto A movimentado em relação aos obstáculos B_j gerando os novos obstáculos $CO_A(B_j)$. O método de representação espacial sugerido por Lozano-Perez é melhor que os usados por Lee e Hoel, uma vez que não é necessário gerar um número muito grande de celas para obter um grau de precisão acurado. Por outro lado a introdução de objetos poliédricos convexos permite uma representação mais adequada do mundo real que a do tradicional "mundo de blocos" usada na área de inteligência artificial. A presença de objetos de forma mais complexas não convexas pode ser resolvida mediante a decomposição do objeto em formas convexas manipuláveis pelo método. Isto pode acarretar que, numa configuração com relativamente poucos objetos o trabalho de gerar os $CO_A(B_j)$ torne-se demorado devido à multiplicação do número de objetos na configuração. Por outro lado, no caso de existirem objetos com contornos curvos, uma forma simplificada de tratamento seria fazer uma sobre-estimação bastante simples do contorno do objeto usando o menor número possível de traços retos, de modo a evitar a geração de um

número muito grande de arestas o que acarretaria ineficiência computacional. Um efeito secundário desta abordagem seria uma diminuição do espaço livre real existente.

A técnica de representação geométrica detalhada por parte destes métodos não anula a possibilidade de fracassos durante a busca do caminho, pois geralmente as situações do mundo real estão sujeitas a imprecisões nos dados referentes à posição dos obstáculos e à forma exata destes e do objeto movimentado.

O método de BROOKS (1983), procura obter uma "folga" entre o objeto movimentado e os obstáculos, que é geralmente maior que no método de Lozano-Perez. Adicionalmente, os casos em que acontecem rotações do objeto movimentado são tratados de maneira mais simples e eficiente que em LOZANO-PEREZ (1982). O método de Brooks consiste (para o caso de objetos poligonais) em gerar cones generalizados entre cada par de obstáculos adjacentes (Ver fig. II.6). A geração de cones desta maneira geralmente implica em sobreposição parcial destes (O método de Lozano-Perez também inclui a possibilidade de sobreposições dos $CO_A(B_j)$). Para cada interseção de cones é computada a faixa de orientações válidas para o objeto movimentado. Cada cone tem um corte transversal chamado "spin".

Os arcos no grafo de busca correspondem a se transferir de um ponto de interseção no spin para outro (translação), e àqueles que correspondem a se transferir de um spin para outro no seu ponto comum de interseção. (Mudança de orientação do objeto). Todos os pares de nós candidatos são checados para verificar sua conectividade. E

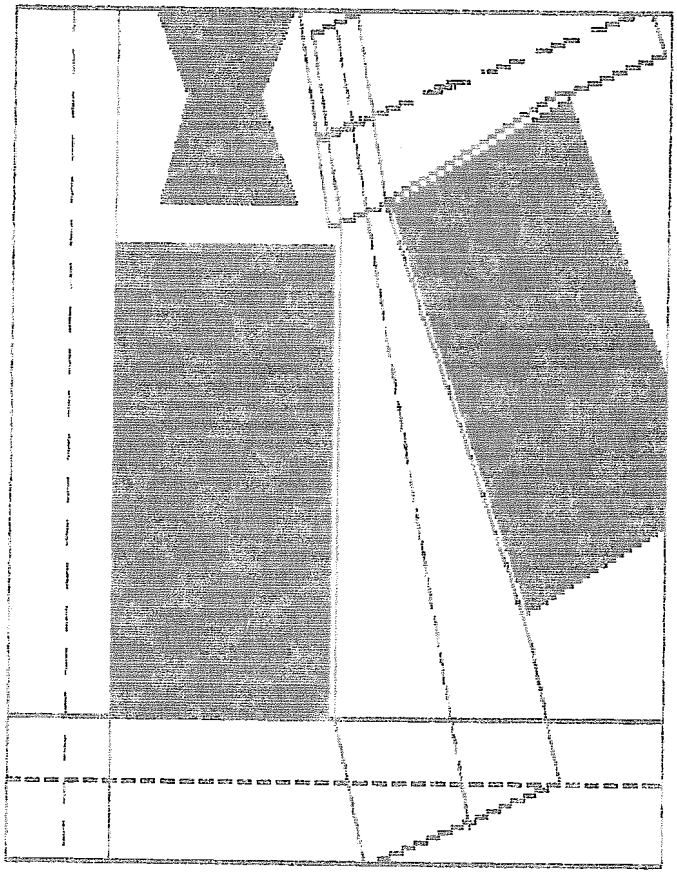
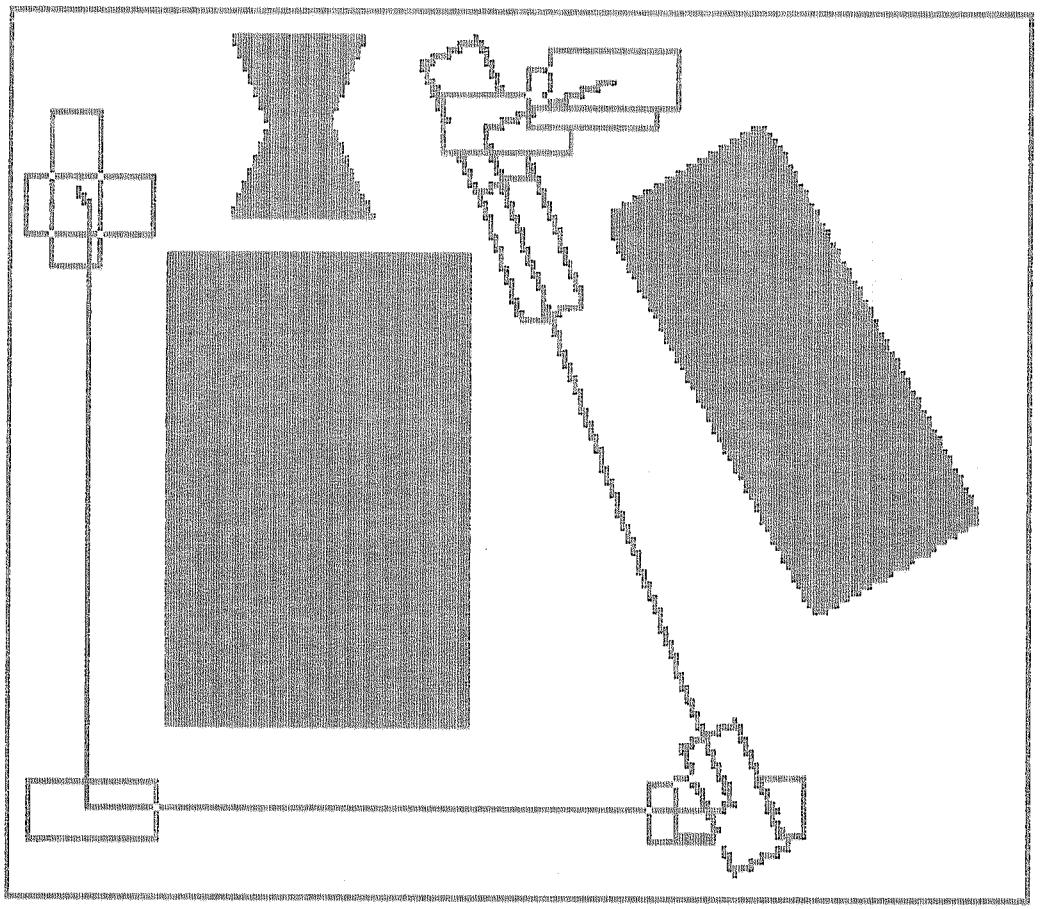


FIG. 11.6

condição suficiente para garantir a conectividade, verificar para os arcos intracones se os intervalos de orientações dos dois nós possuem interseção não nula; e para os arcos intercones verificar se a interseção de orientações válidas entre o par de nós não é vazio, uma vez que não há mudança de rotação durante a translação. O objeto é movido seguindo a linha do spin de cada cone generalizado no caminho determinado. A quantidade de computação requerida é independente do sistema de coordenadas utilizado. A busca é feita utilizando o A* [NILSSON (1980)].

Este método tem fraco desempenho na determinação de caminhos quando a configuração espacial é muito compacta, uma vez que o número de cones generalizados não é geralmente suficiente para gerar um número variado de caminhos viáveis. Em ambientes relativamente pouco densos, o algoritmo é muito rápido; este método geralmente é aplicado como orientação para outros métodos mais complexos como o de Lozano-Perez. Entretanto, a obrigação do objeto movimentado de se deslocar através dos spins dos cones generalizados pode sacrificar a obtenção de soluções melhores que seriam possíveis se o objeto tivesse liberdade de se deslocar através de outro percurso no espaço livre que não obrigatóriamente o dos spins. A complexidade envolvida na geração dos cones generalizados para o caso de poligonos é da ordem de $O(n^3)$ onde n é o número total de faces dos obstáculos. A utilização na busca do caminho do algoritmo A* possibilita uma busca mais seletiva no grafo de espaço livre sem no entanto afetar a complexidade inerente ao problema que é exponencial (Ver fig. II.7).

O uso de heurísticas sobre o problema pode



CAMINHO ACHADO PELO ALGORITMO

FIG. II.7

tornar o algoritmo mais eficiente; as heurísticas geralmente usadas nestes métodos restringem-se a heurísticas relativas ao tempo para percorrer um caminho e à métrica do caminho determinado. Estas heurísticas tem aplicação permanente durante a busca e não é possivel a incorporação de novas heurísticas durante a execução do algoritmo. O desempenho do A* geralmente é medido através de estatísticas que comparam o número de nós abertos e número de nós fechados. Como nos métodos que geram celas como forma de representação espacial o número destas é conhecido e fixo, também podemos incluir este número nas estatísticas comparativas do desempenho do algoritmo de busca pois representam o número total de nós do grafo.

Uma heurística (no sentido geral e não apenas o usado no algoritmo A*) correntemente usada neste tipo de problema é a de gerar uma representação detalhada apenas nos pontos próximos dos obstáculos, e representação simplificada longe destes.

As restrições comumente consideradas nestes métodos dizem apenas respeito à proibição de colisões, não sendo consideradas outro tipo de restrições que surgem ocasionalmente no mundo real como proibição de passagem por áreas proibidas, distância mínima a ser mantida em relação a um objeto particular da configuração (p. ex. no caso de objetos de temperatura elevada) e outras que serão mencionadas posteriormente. A incorporação de um número variado de restrições , índices de performance e conhecimento sobre condições do problema, obriga a uma revisão completa do método a ser aplicado para cada situação particular.

Uma análise comparativa das principais características dos métodos apresentados neste capitulo é feita a seguir.

a) Aplicação Principal.

Lee : diagramação elétrica, desenhos lógicos.

Hoel : diagramação elétrica, desenhos lógicos.

Lozano-Perez : busca do caminho para objeto móvel. Robótica.

Brooks : busca do caminho para objeto móvel. Robótica.

b) Tipo de obstáculos considerados.

Lee : arestas, retângulos com arestas alinhadas com os eixos coordenados.

Hoel : arestas, retângulos com arestas alinhadas com os eixos coordenados.

Lozano-Perez : poliédricos, sem restrição quanto à orientação.

Brooks : poliédricos, sem restrição quanto à orientação.

c) Representação espacial.

Lee : celas quadradas homogêneas.

Hoel : celas quadradas homogêneas.

Lozano-Perez : VGRAPH, celas poliédricas.

Brooks : celas cônicas.

d) Nivel de precisão da representação espacial.

Lee : detalhado.

Hoel : detalhado.

Lozano-Perez : detalhado.

Brooks : pouco preciso, serve de orientação para métodos mais precisos.

e) Características do algoritmo de busca.

Lee : ineficiente, aceita número limitado de restrições e índices de performance mas conduz a soluções ótimas.

Hoel : mais eficiente que o de Lee. Aceita número limitado de restrições e índices de performance. Conduz a soluções ótimas.

Lozano-Perez : eficiente, aceita número limitado de heurísticas, restrições e índices de performance. Baseado no algoritmo A* que sob certas condições conduz a soluções ótimas.

Brooks : eficiente, aceita número limitado de heurísticas, restrições e índices de performance. Baseado no algoritmo A*. Geralmente usado como orientação para métodos mais precisos.

CAPITULO III - REPRESENTAÇÃO DO CONHECIMENTO : METODO DE YASUHIRO

A necessidade de uma estratégia global na solução de problemas de planejamentos de rotas, que possibilite a incorporação de um número grande e variado de restrições e índices de performance alem de conhecimento sobre o problema em geral, é tratada por YASUHIRO e WADA (1986) através de um método baseado em representação e utilização do conhecimento para o planejamento de rotas. A aplicação desse método restringe-se a configurações onde os obstáculos são objetos retângulares e alinhados com os eixos coordenados e o objetivo consiste em estabelecer ligações entre pares de pontos na configuração dada. Não é considerado o caso do planejamento da rota para um objeto móvel de dimensões definidas, nem quando este permanece com orientação fixa durante todo o percurso, nem quando ocorrem rotações. O tipo de rota determinado obedece à restrição de também ser alinhado com os eixos coordenados.

O método de busca baseado em conhecimento sugerido por Yasuhiro procura encontrar a rota ótima de acordo com os índices de performance, restrições e heurísticas especificadas pelo projetista. O método é baseado em tres técnicas chaves; a primeira é a representação do espaço livre do layout por celas obstáculo-adaptativas, a segunda é a aplicação direta e automática do conhecimento sobre rotas no processo de busca por meio de computação simbólica. Uma especificação detalhada da rota a ser planejada é obtida por inferência do conhecimento sobre rotas contido na base de conhecimento. A

terceira técnica é a extensão de um algoritmo de determinação de rotas em labirintos utilizando o algoritmo A* como técnica de busca que permite a incorporação da nova representação do espaço livre e a aplicação flexível da busca baseada em conhecimento.

O método pode ser implementado em três módulos básicos : módulo de redução do conhecimento, módulo de geração da representação do conhecimento através de frames e módulo de busca do caminho. A interação entre os módulos é ilustrada na fig. III.1. O módulo de redução do conhecimento utiliza uma função de inferência para obter, a partir da base de conhecimento, o conhecimento denominado de primitivo. Dois tipos de conhecimento são utilizados; um tipo compreende heurísticas para o processo de resolução do problema e o outro compreende restrições e índices de performance. O módulo de geração da representação de frames é o que gera os frames da rota e obstáculos; ele carrega o conhecimento primitivo obtido no módulo de redução do conhecimento e preenche os slots adequados dos frames associados à rota e obstáculos. O módulo de busca do caminho compreende as etapas de entrada/saída de dados do programa de planejamento de rotas, geração de celas obstáculo-adaptativas e determinação da rota ótima. Unicamente por fins didáticos, o método é apresentado para o caso de duas dimensões. (Eixos X-Y).

Nas seções seguintes serão examinadas as técnicas usadas para representação do espaço livre, utilização do conhecimento sobre rotas e o determinação da rota ótima.

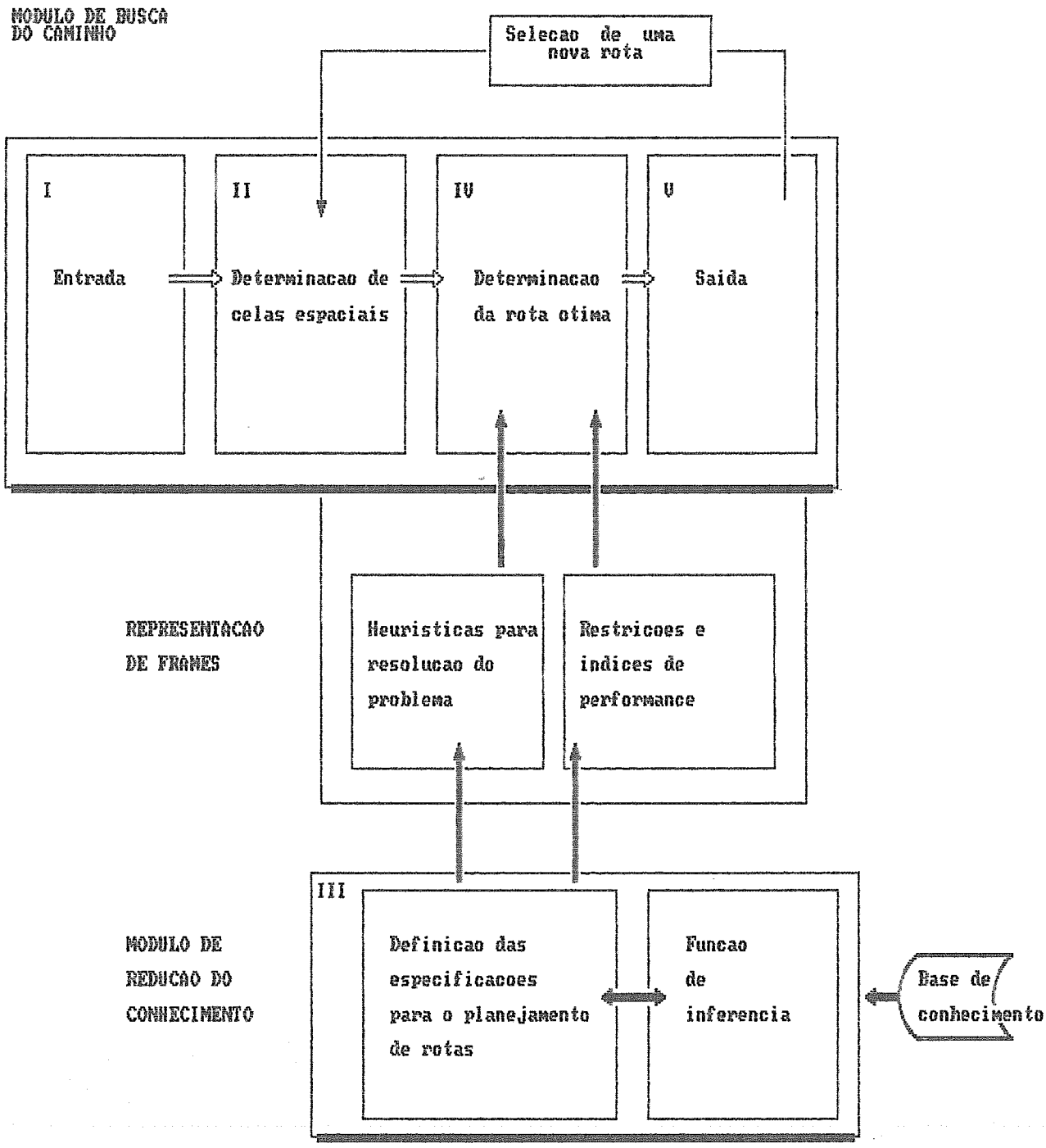

MODULO DE BUSCA
DO CAMINHO
Selecao de uma
nova rota
I
Entrada
II
Determinacao de
celas espaciais
IV
Determinacao
da rota otima
V
Saida
REPRESENTACAO
DE FRAMES
Heuristicas para
resolucao do
problema
Restricoes e
indices de
performance
III
MODULO DE
REDUCAO DO
CONHECIMENTO
Definicao das
especificacoes
para o planejamento
de rotas
Funcao
de
inferencia
Base de
conhecimento

FIGURA III.1

3.1 Representação do espaço livre.

O espaço livre é gerado de forma explicita, à diferença do método de Lozano-Perez, que procura representar com bastante detalhe somente o espaço livre próximo dos obstáculos. O método de geração de celas, de Yasuhiro, representa com igual nível de detalhe todo o espaço livre da configuração espacial.

Como os obstáculos são retângulares e alinhados com os eixos coordenados, as celas geradas também possuem esta mesma característica; alem disso, suas dimensões dependem das dimensões dos obstáculos, por isso são chamadas ostáculo-adaptativas. As celas obstáculo-adaptativas são definidas de tal maneira que cada cela possa colidir hipotéticamente com um único obstáculo caso esta se movimente paralelamente a qualquer um dos eixos coordenados. Dito com outras palavras, cada cela pode "avistar" um único obstáculo em qualquer uma de suas quatro direções. Este tipo de representação permite descrever a topologia da configuração com um mínimo de informação.

O método de geração de celas consiste de três etapas bem definidas : geração, teste e combinação. Dada uma configuração, as celas são obtidas prolongando-se as arestas dos obstáculos até o contorno da configuração. Cada retangulo obtido pelas interseções desses prolongamentos é, em principio, uma cela. Na etapa de teste é verificado se alguma cela assim gerada intersecta com algum obstáculo; caso positivo esta é excluida da lista de celas. Na etapa da combinação é verificado se duas celas vizinhas "avistam" os mesmos obstáculos nas quatro direções possíveis. Caso positivo estas são combinadas para formar uma única cela. As

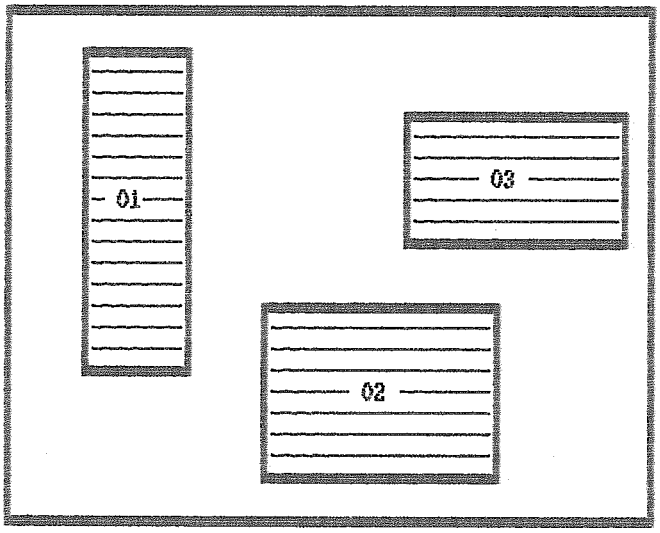
figuras III.2b, III.2c e III.2d ilustram cada uma das etapas do processo de geração de celas para a configuração exemplo da figura III.2a.

3.2 Utilização do conhecimento sobre rotas.

No inicio deste capitulo falamos sobre a manipulabilidade de índices de performance, restrições e heurísticas variados permitidos por este método. O módulo de busca do caminho aceita um número variado de restrições e índices de performance. Se a informação é dada neste nivel primitivo, o módulo de busca aceita estas diretrizes e busca um caminho compativel com estas. Se a informação é dada em um nivel mais alto, é necessária uma tradução para um nível mais baixo para que seja aplicável na busca.

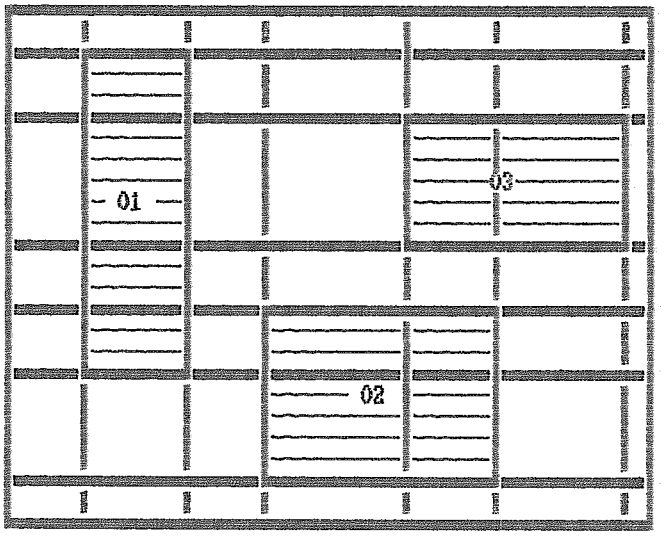
A definição do problema a ser resolvido no módulo de busca do caminho é definida atraves da interação entre o módulo de redução do conhecimento e a base de conhecimento. Esta interação está baseada no processo de inferência para obter conhecimento primitivo sobre uma rota através de dialogo com a base do conhecimento.

A base de dados possui conhecimento sobre planejamento de rotas na forma de regras de produção (Se - então) e frames. Este conhecimento deve englobar aspectos tais como que parte do layout está disponivel para a rota, quais são os via-points da rota (Os via-points são pontos pelos quais a rota deve obrigatóriamente passar), quais os tipos de restrições existentes e os seus possiveis valores limite, quais os índices de performance e quais os pontos inicial e final da rota a ser determinada.



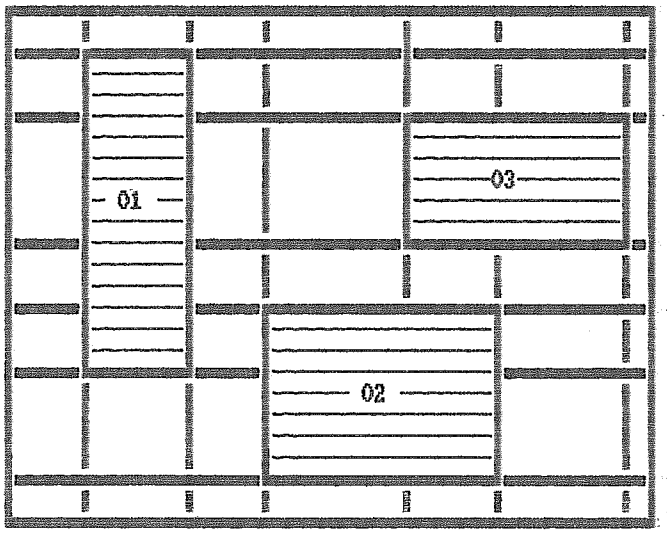
Configuracao problema contendo limites do layout
e obstaculos

FIGURA III.2a



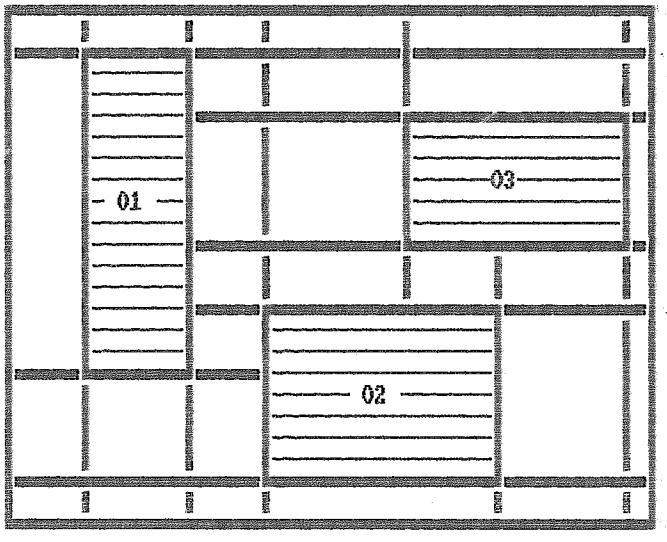
Celas geradas inicialmente na etapa de geracao

FIGURA III.2b



Situacao das celas depois da etapa de teste

FIGURA III.2c



Situacao das celas depois da etapa de combinacao

FIGURA III.2d

3.2.1 Conhecimento diretamente aplicável na busca do caminho.

Existem duas etapas a serem aqui realizadas : (a) redução do conhecimento para a forma primitiva tal como restrições e índices de performance e (b) aplicação deste conhecimento primitivo na determinação da rota. Se o conhecimento é fornecido em um nível primitivo, o módulo de busca do caminho aplica este conhecimento diretamente na busca da rota. Se por outro lado, o conhecimento é dado em um nível conceitual mais elevado, é necessário antes uma tradução para a forma primitiva. Na tabela III.1 são mostrados alguns exemplos de conhecimento primitivo.

TABELA III.1

Tipo	Conhecimento Primitivo	Exemplo
Decomposição Serial da rota	Via-Points	A rota deve passar pelos pontos A e B nessa ordem
Restrições	Area Proibida	A rota não deve passar pela área A.
	Direção Proibida	A rota é proibida de seguir na direção Norte.
	Distancia Mínima de obstáculos	A rota é proibida de se aproximar a menos de 1 metro do obstáculo 03.
	Tamanho mínimo do segmento	A rota é proibida de ter um segmento menor que 10 metros.
Indices de performance	Tamanho da rota	A rota de menor tamanho deve ser preferida.

A representação do conhecimento utiliza frames, onde os slots típicos a serem preenchidos são no caso da rota : Inicio (cela inicial), Objetivo (cela objetivo), Tipo de rota (p. ex. sensivel-ao-calor), Largura da rota, Parte (nome dos frames de rota "filhos"), Parte-de (nome do frame "pai"), via-points (nome das celas que constituem via-points), Pontos de referência (nome de celas que são pontos-de-referência), Restrição (nome das restrições a serem aplicadas à rota),Distancia "default" (da rota para os obstáculos e limites do layout), Distancia-mínima (da rota para obstáculos e limites do layout específicos), Tamanho do segmento, Area proibida (nome de celas), Direção proibida , Indices de Performance, Tamanho total, Solução (sequência de labels de celas).
No caso dos frames de obstáculos, os slots típicos são Nome e Tipo-de-obstaculo.

A redução do conhecimento ("tradução") é feita em duas sub-etapas. P. ex. no caso da distancia-minima o procedimento é o seguinte : O módulo de redução do conhecimento gera uma pergunta para o módulo de inferência :

E a proposição ("distancia-minima" Rota-1 x) verdadeira ?

A resposta pode ser obtida na seguinte forma :

Verdade, e o valor de x é igual a (default 1) ou (01 8)

Esta resposta sugere duas proposições :

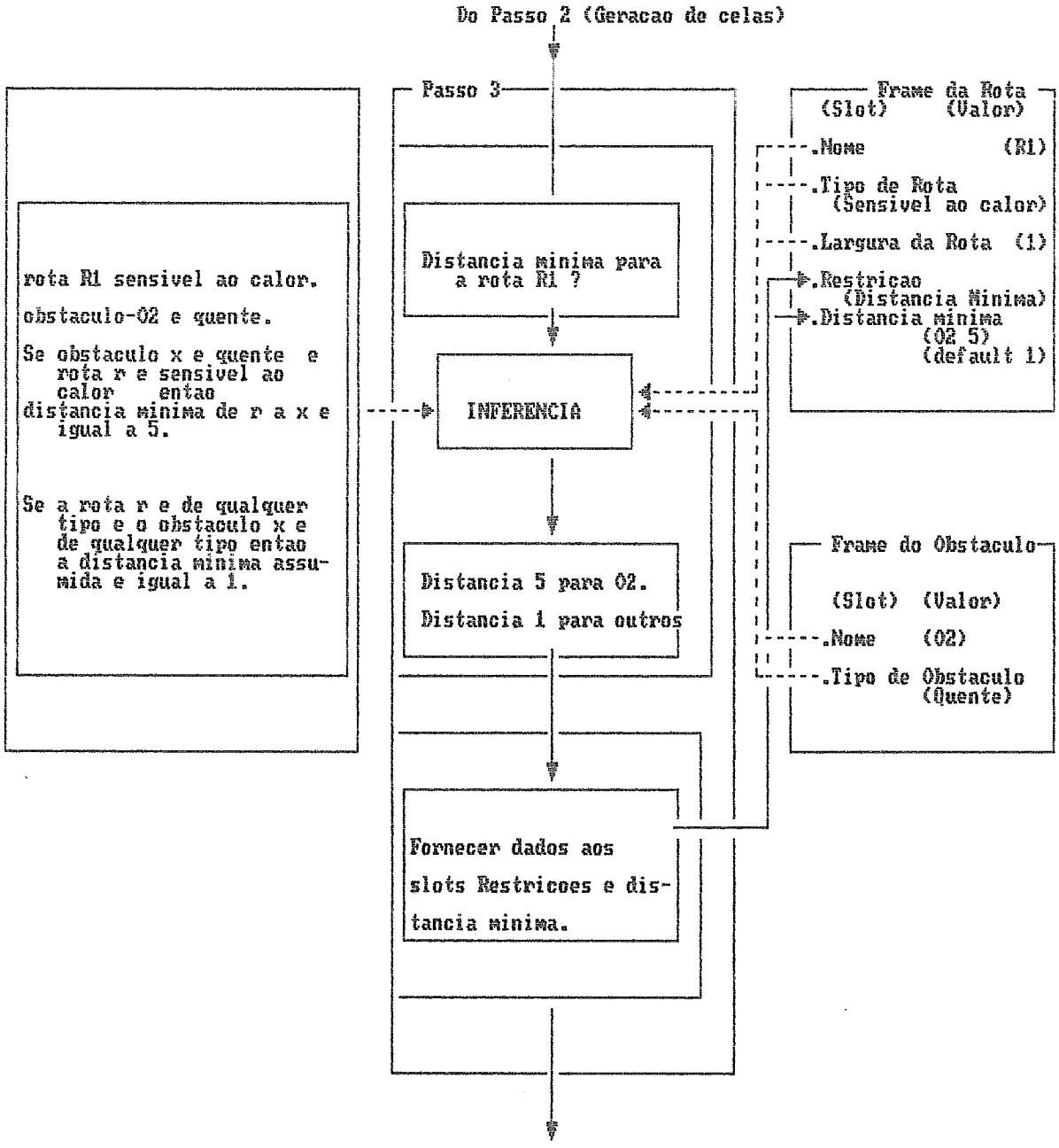
("distancia-minima" Rota-1 (default 1)) e

("distancia-minima" Rota-1 (01 8))

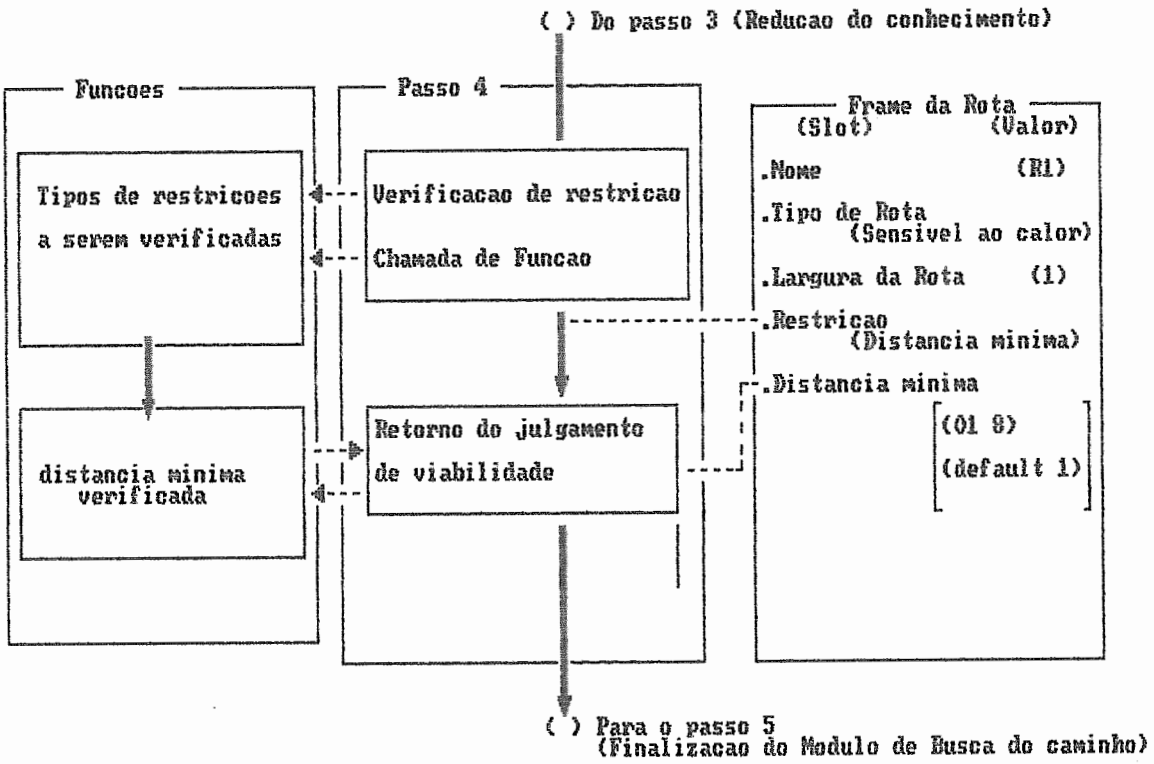
A primeira proposição indica que a distancia minima de uma rota 1 em relação a um obstáculo qualquer é assumida como de valor igual a 1 quando inexiste uma informação especifica determinando a distancia mínima a ser mantida em relação ao obstáculo. A segunda proposição determina que a distancia minima da rota 1 em relação ao obstáculo 1 é de valor igual a 0.

Na segunda sub-etapa são preenchidos no frame da rota os slots "restrição" e "distancia-minima" com as informações obtidas. Este procedimento é repetido até obter suficientes informações sobre as especificações da rota. O esquema da fig. III.3 ilustra o mecanismo de redução do conhecimento.

A utilização do conhecimento primitivo no módulo de busca do caminho é realizada em duas sub-etapas : a primeira sub-etapa invoca a função para verificação de restrições, que identifica o tipo de restrições a serem aplicados à rota. Cada restrição que aparece no slot "Restrição" do frame da rota possui uma função correspondente que fará a avaliação da restrição para a cela que é candidata a conter um segmento de rota e que foi gerada pelo algoritmo de busca do caminho durante a determinação da rota ótima. A segunda sub-etapa é aceitar o julgamento de viabilidade da cela a ser avaliada pela função de verificação de restrições invocada na primeira sub-etapa. O conhecimento primitivo contido no frame serve tanto para definir os indices de performance a serem aplicados à rota como as restrições existentes em uma forma modularizada. O esquema da fig. III.4 ilustra a utilização do conhecimento primitivo.


Do Passo 2 (Geracao de celas)
Passo 3
Frame da Rota
(Slot) (Valor)
.Nome (R1)
.Tipo de Rota
(Sensivel ao calor)
.Largura da Rota (1)
.Restricao
(Distancia Minima)
.Distancia minima
(02 5)
(default 1)
Distancia minima para a rota Ri ?
INFERENCIA
rota R1 sensivel ao calor.
obstaculo-02 e quente.
Se obstaculo x e quente e rota r e sensivel ao calor entao distancia minima de r a x e igual a 5.
Se a rota r e de qualquer tipo e o obstaculo x e de qualquer tipo entao a distancia minima assumida e igual a 1.
Distancia 5 para 02.
Distancia 1 para outros
Frame do Obstaculo
(Slot) (Valor)
.Nome (02)
.Tipo de Obstaculo
(Quente)
Fornecer dados aos slots Restricoes e distancia minima.
Para o Passo 4 (Determinacao da rota otima)

F I G U R A III.3


( ) Do passo 3 (Reducao do conhecimento)
Funcoes
Passo 4
Frame da Rota
(Slot)
(Valor)
Tipos de restricoes
a serem verificadas
Verificacao de restricao
Chamada de Funcao
.Nome
(R1)
.Tipo de Rota
(Sensivel ao calor)
.Largura da Rota
(1)
.Restricao
(Distancia minima)
.Distancia minima
(01 8)
(default 1)
distancia minima
verificada
Retorno do julgamento
de viabilidade
( ) Para o passo 5
(Finalizacao do Modulo de Busca do caminho)

F I G U R A III.4

3.2.2 Conhecimento sobre Solução de problemas.

O conhecimento de heurísticas para resolução do problema, se existente, deve ser amplamente utilizado pois possibilita um aumento da eficiência computacional. Aqui é explorada a possibilidade de decomposição do problema por meio de via-points. Esta decomposição possibilitará um aumento de eficiência do processo de busca. Aqui também são necessária duas etapas : redução do conhecimento para conhecimento primitivo e utilização do conhecimento primitivo. A decomposição da rota é descrita como um exemplo de heurística para resolução de problemas para explicar as duas sub-etapas da utilização do conhecimento.

a) Redução do conhecimento sobre a rota para conhecimento primitivo.

Aqui informações como p. ex.: "A rota da tubulação deve passar entre dois obstáculos específicos que podem ser equipamentos ou estruturas", servem para obter informações sobre como decompor em rotas menores utilizando "via-points". O conhecimento de via-points se disponivel, deve ser maximamente utilizado para decompor o problema original em sub-problemas para aumentar a eficiência computacional.

b) Utilização do conhecimento primitivo.

O slot de "via-points" no frame da rota é preenchido com n nome de celas, e os seguintes passos são executados.

b.1) (n + 1) frames "filhos" da rota são gerados a partir do frame "pai", e os slots "inicial", "objetivo" e "Parte de" são preenchidos com os valores correspondentes. Os outros

valores dos slots do frame são "herdados" do frame "pai".

b.2) O módulo de busca do caminho é aplicado para cada sub-rota e o slot "solução" é preenchido com o valor determinado.

b.3) As soluções dos frames "filhos" são sintetizados para produzir a solução do frame "pai".

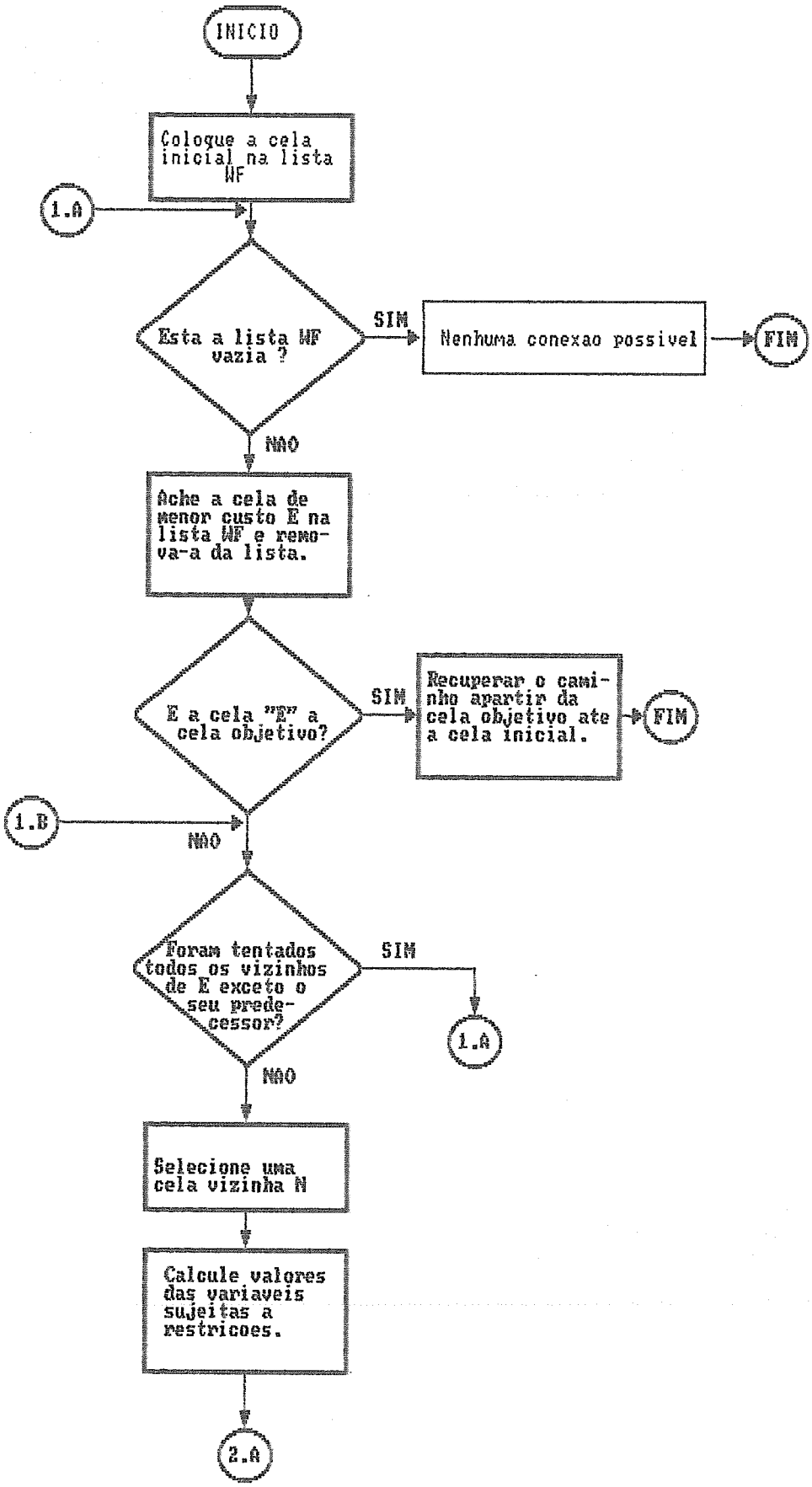
3.3 Determinação da Rota.

A determinação da rota pode ser feita usando uma versão especializada do A* no qual pode ser feito um controle explicito do processo de enumeração de celas. O algoritmo está descrito no fluxograma das figs. III.5a e III.5b.

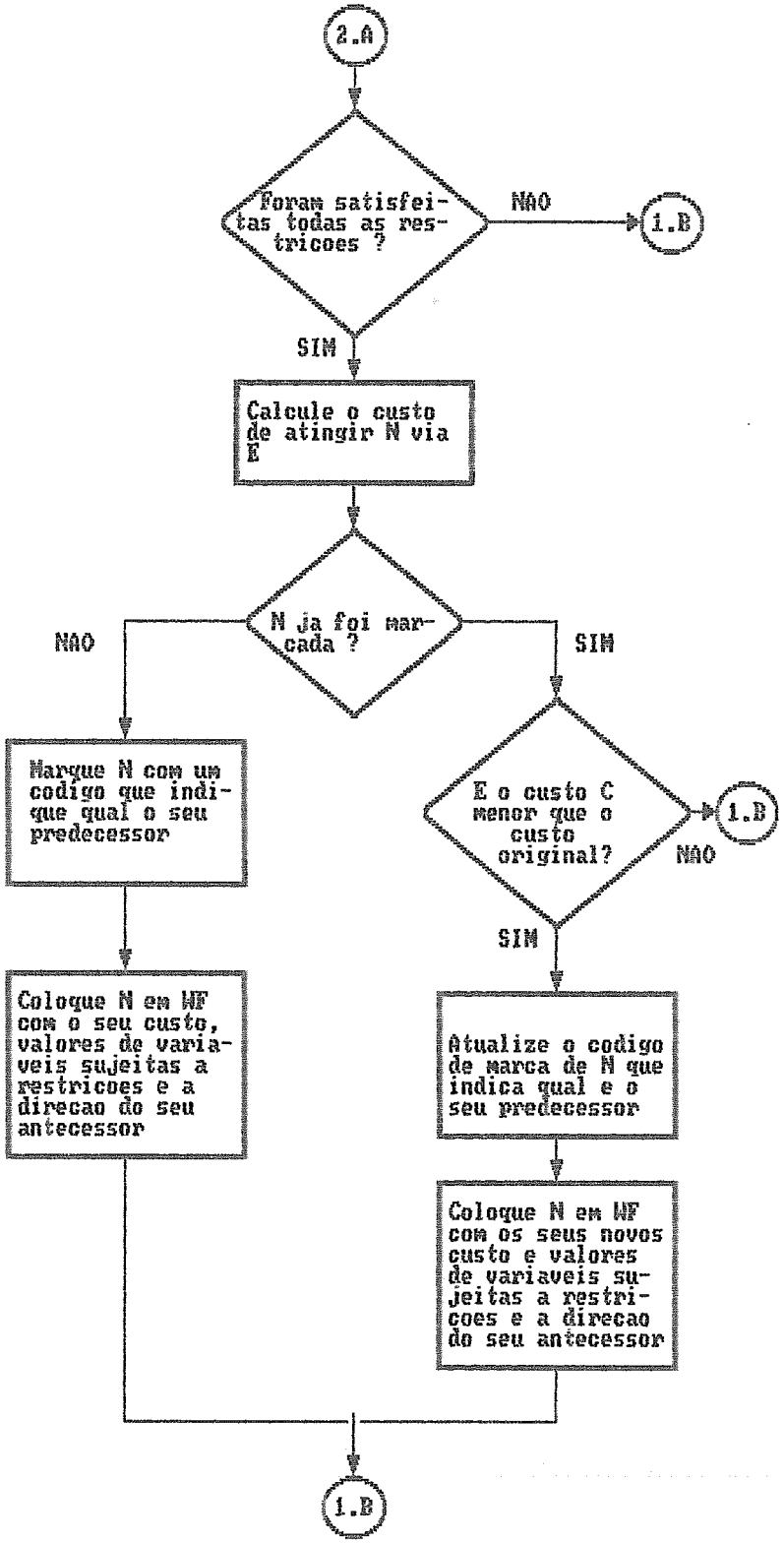
3.4 Análise do Método de Planejamento de Rotas.

Existem algumas questões que procuraremos analisar e dar uma resposta satisfatória. No módulo de busca do caminho por exemplo, é assumido que atingir uma nova cela no caminho implica em estabelecer uma ligação entre o centroide da cela que está sendo expandida e o da que foi atingida. Isto permite que uma cela qualquer tenha a capacidade de abrigar até dois segmentos de rotas diferentes e perpendiculares entre si, se for permitido o cruzamento de rotas. O método de planejamento de rotas não nos diz porém como proceder em caso de que se queira explorar ao máximo o espaço livre dentro da cela para permitir a passagem do maior número de rotas possível, por ela.

Outra pergunta diz respeito a um aperfeiçoamento do processo de utilização do conhecimento. Esta questão surge especificamente no passo do algoritmo de busca que testa a viabilidade das celas. Como uma mesma cela



F I G U R A III.5a



M O D U L O D E B U S C A D O C A M I N H O

F I G U R A III.5b

pode ser atingida várias vezes durante a busca do caminho, seria interessante se pudermos reaproveitar pelo menos algumas informações obtidas anteriormente. Como a existência de restrições para um problema como o nosso estabelece uma linha divisória para o traçado de segmentos de rota, assim temos para uma rota até então determinada, expansões viáveis e não viáveis. No entanto, existirão algumas situações em que quebrar algumas restrições pode-se revelar proveitoso para obter um caminho muito melhor do que seria obtido se fossem respeitadas todas as restrições ou para evitar uma condição de fracasso na busca. É claro que na prática nem todas as restrições se revelarão possíveis de serem quebradas. Cabe então acrescentar ao conhecimento sobre o problema, a informação de quais as restrições que podem ser quebradas, e efetuar modificações no módulo de avaliação de restrições. Estas modificações serão descritas no capitulo VI. Outras questões a serem respondidas são : O uso de técnicas de inteligência artificial ajuda a melhorar a qualidade da solução obtida em relação a outros métodos? O A* especializado aqui utilizado é o melhor algoritmo do qual podemos dispor para realizar a busca? Como determinar a eficiência das heurísticas utilizadas? E quanto à representação do conhecimento, porque o uso de frames e regras de produção? Como tratar os casos de incerteza na configuração, comuns nos problemas reais ? uma indagação final seria se o método de planejamento de rotas poderia ser incorporado por outros métodos para resolução de problemas mais complexos do mundo real. Tais questões serão abordadas nos capítulos seguintes.

CAPITULO IV - TOPICOS DE INTELIGENCIA ARTIFICIAL E O PROBLEMA DE PLANEJAMENTO DE ROTAS.

Neste capitulo serão dados os fundamentos teóricos necessários à melhor compreensão das técnicas de inteligência artificial utilizadas neste trabalho e a sua aplicação no método de planejamento de rotas. Inicialmente será feita uma explanação sobre a natureza do processo de resolução de problemas e as principais estratégias usadas. A seguir analisaremos o problema do planejamento de rotas no contexto do mundo real. A conceituação e justificação das estruturas de representação do conhecimento utilizadas assim como a descrição do que entendemos por Sistemas de Produção também serão abordadas. Por último serão descritos de forma resumida, os principais atributos da linguagem LISP, a linguagem na qual foi implementado o Sistema de Produçao de Planejamento de Rotas. O Módulo de Inferência de nosso Sistema será descrito à parte devido à complexidade envolvida.

4.1 Resolução de Problemas.

NEWELL (1969), classificou os problemas em dois tipos : Os bem esturuturados e os fracamente estruturados ou, análogamente, os formalizáveis e os não formalizáveis. A formalização do problema implica sempre no primeiro tipo. Por problema bem estruturado ele se refere ao problema cujo tratamento pode ser formalizado, e para o qual existe no mínimo um método que representa uma solução geral para o problema. Os problemas bem estruturados devem tambem possuir um objetivo bem definido e o método existente de

resolução deve ser capaz de obter uma solução para quaisquer que sejam os dados de entrada. Os problemas fracamente estruturados são definidos pelo negativo : são todos os problemas que não são bem estruturados. Na realidade parece não haver uma linha divisória nítida entre estes dois tipos de problemas; de um lado estão os problemas cujos métodos de resolução estão bem definidos assim como os seus domínios de aplicação; de outro lado estão o restante dos problemas cujos métodos de resolução não são completamente formalizáveis ou sobre os quais pouco se conhece. A formalização de um método de resolução de problemas implica, entre outros aspectos, no conhecimento completo do domínio de aplicação e da natureza do problema. Os métodos fracamente estruturados demandam menos informação do ambiente de aplicação que os bem estruturados (métodos formais), enquanto estes últimos se tornam cada vez menos gerais na medida do aumento da demanda de informação, mas por outro lado são os que tem melhor poder pois tendem a apontar para soluções ótimas. O poder de um método também é avaliado pela capacidade deste em obter uma solução para dados de entrada quaisquer, para um problema do seu escopo. A figura IV.1 extraida de NEWELL (1969) ilustra o anteriormente exposto.

Cada ponto do gráfico está associado com um método diferente. A abcissa representa aumento de informação na declaração do problema, isto é, sua decrescente generalidade. A ordenada representa o crescimento de poder. Para cada grau de generalidade existe um limite superior no possível poder do método. A posição de alguns pontos afastada da bisetriz ilustra a hipótese de Newell de que há

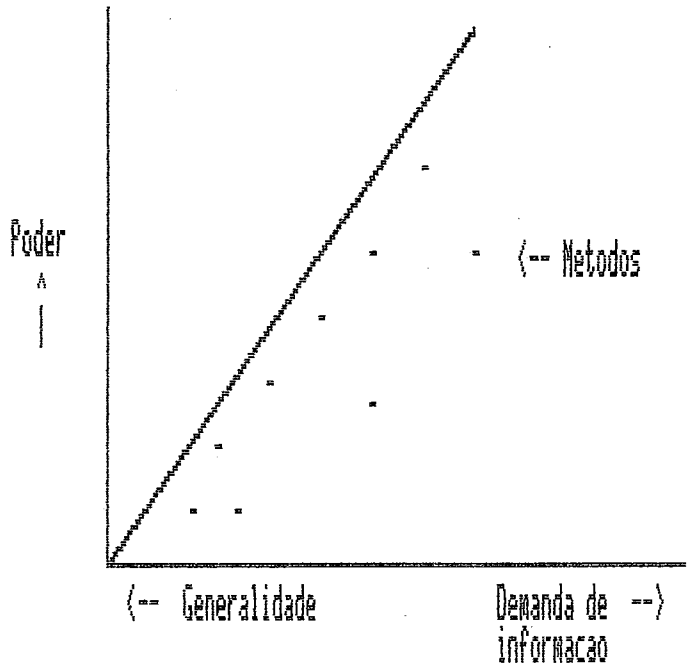


FIGURA IV.1

métodos que não exploram completamente a informação contida na definição do problema. O fato de alguns problemas serem classificados agora como fracamente estruturados não quer dizer que permanecerão sempre nesta categoria. Estes problemas podem chegar a ter uma formalização completa no futuro. Um exemplo pode ser dado com os métodos de programação linear. A aplicação do método simplex, por exemplo, em planejamento pode produzir soluções ótimas. Não obstante, antes da definição deste método, a solução de problemas do escopo da PPL pelo planejador consistia em métodos pouco precisos usados em conjunto com a sua experiência e habilidade pessoal. Por outro lado, a existência de algoritmos bem definidos, que antes era privilégio dos problemas classificados como bem estruturados, passou a ser também uma característica de alguns tipos de problemas considerados fracamente estruturados, como por exemplo os problemas solúveis por algoritmos heurísticos, e mais genéricamente falando, de alguns problemas do escopo da área de Inteligencia Artificial, embora a completude do método de resolução nem sempre seja garantida. Uma discussão mais ampla da classificação de problemas em bem estruturados e em fracamente estruturados pode ser achada em NEWELL (1969).

Podemos considerar como sendo os principais passos na resolução de problemas :

a) Definição / Representação do Problema.

b) Geração do plano de solução.

c) Implementação deste plano.

Na representação do problema está envolvida

a manipulação simbólica da representação deste em uma forma equivalente mas que se revela mais útil que a anterior. Os provadores de teoremas p. ex. utilizam tais manipulações. O ser humano possui a capacidade de lidar tanto com problemas fraca como fortemente estruturados, e é esta capacidade que lhe permite a transformação dos primeiros nos segundos.

Na representação de problema também incluimos o conhecimento que possuimos sobre o problema, tais como a modelagem do ambiente, a situação inicial e a situação objetivo.

A área de inteligência artificial tem se caracterizado pelo desenvolvimento de estratégias de solução para problemas ainda não completamente formalizáveis, fracamente estruturados. Para resolução de determinados problemas, como por exemplo, os que surgem na área de Engenharia, tem sido aplicadas estratégias de solução que combinam métodos de I.A. com métodos formais.

4.1.1 Sistemas de Produção.

Os programas de inteligência artificial são comumente estruturados segundo uma arquitetura que facilita a descrição do processo de busca da solução. Esta arquitetura recebe o nome de Sistema de Produção. Um Sistema de Produção possue básicamente tres componentes.

a) Uma ou mais bases de dados que contem informações apropriadas para uma tarefa particular. Algumas partes da base de dados podem ser permanentes, enquanto outras podem pertencer apenas a solução do problema corrente.
b) Um conjunto de regras, cada uma consistindo de uma lado esquerdo (LHS) que determina sua aplicabilidade e um lado

direito (RHS) que descreve a ação a ser executada.

c) Uma estratégia de controle que especifica a ordem em que as regras serão comparadas com a base de dados e a maneira de resolver os conflitos que possam surgir quando várias regras são aplicáveis ao mesmo tempo.

Há várias diferenças entre a estrutura de um Sistema de Produção e um Sistema convencional em programas organizados hierarquicamente. Em um Sistema de Produção a base de dados global pode ser acessada por todas as regras. As regras não chamam outras regras; a comunicação entre regras ocorre somente através da busca de dados global. Um dos problemas em utilizar sistemas convencionais, de programas organizados hierarquicamente, em aplicações em I.A. é que adições ou mudanças na base de conhecimento podem requerer grandes mudanças nas estruturas de dados e organização de sub-rotinas. O projeto de Sistemas de Produção é muito mais modular e mudanças na base de dados, ou no mecanismo de controle ou nas regras podem ser feitas de maneira relativamente independente. Na maior parte das aplicações, a informação disponível para a estratégia de controle não é suficiente para permitir a seleção da regra mais apropiada. As operações de um Sistema de Produção podem ser caracterizadas como um processo de busca no qual as regras são tentadas até que alguma sequência delas produza uma base de dados que satisfaça a condição de parada.

No Capítulo V é descrito o Motor de Inferência desenvolvido para nosso Sistema baseado em técnicas eficientes de processos de inferência.

4.1.2 Estratégias de Controle.

Existem duas classes principais de estratégias de controle para Sistemas de Produção : Irrevogável e Tentativas. Num regime de controle irrevogável, uma regra aplicável é selecionada e aplicada irrevogàvelmente sem previsão de reconsideração posterior. Em um regime de controle por tentativas, uma regra aplicável é selecionada (ou arbitrariamente ou devido a uma boa razão), a regra é aplicada, mas uma previsão é feita para retornar posteriormente a este ponto na computação a aplicar alguma outra regra.

Nós distinguimos dois tipos de regime de controle de tentativas : "backtracking" e busca ordenada. No "backtracking", um ponto de retrocesso é estabelecido quando uma regra é selecionada. Se na computação subsequente é encontrada alguma dificuldade em produzir uma solução, o estado da computação reverte para o ponto de retrocesso imediatamente anterior, onde outra regra é aplicada e o processo continua. No controle de busca ordenada, são armazenados em cada passo os efeitos de varias seqüências de regras simultâneamente. Varios tipos de estratégias e procedimentos de busca em grafos são usados neste tipo de controle. A estratégia de controle por tentativas é muito geral e ineficiente. A constante revisão dos passos torna elevada a complexidade da busca, e o custo computacional pode tornar-se proibitivo. O "backtrack" apesar de sua ineficiência em termos de tempo de processamento, é econômico em termos de consumo de memória. Porém os algoritmos de busca ordenada em grafos, apesar de geralmente serem mais eficientes que o "backtrack" em termos de tempo de processamento, são menos eficientes que este em relação à

memória consumida devido as informações armazenadas em cada estado atingido.

As considerações que foram feitas inicialmente sobre a natureza do processo de resolução de problemas, ajuda-nos a entender o porque da ineficiência destes métodos. A incorporação de conhecimento sobre o problema no processo de busca da solução nos levará a um acrescimo de poder e eficiência. Por exemplo, a posse de conhecimento local infalível na busca irrevogável nos levaria diretamente á solução, seja qual for a configuração inicial de partida, e a tornar explicito o conhecimento global sobre a solução.

Uma forma de ver o processo de busca de uma solução é como o percurso a ser feito em um grafo direcionado, em que cada nó representa um estado do problema e cada arco representa a relação entre os estados representados pelos nós conectados por este.

Algumas considerações importantes sobre o processo de busca aplicáveis a qualquer estrategia são as seguintes :

a) A direção em que deve se conduzir a busca
b) A topologia do processo de busca
c) Como cada nó do processo de busca será representado
d) Seleção das regras aplicáveis
e) Uso de uma função heuristica para guiar a busca

Há duas direções em que se pode realizar o processo de busca : Para-frente ou progressivo, a partir dos estados iniciais, e para-trás ou regressivo, a partir dos estados objetivos. As mesmas regras podem ser usadas para o

raciocinio "para-frente" ou "para-tras" a partir do estado objetivo.

No Calculo de Predicados abordado mais adiante, no raciocinio para-frente, as pre-condições são comparadas com o estado corrente e os lados esquerdos ou conseqüentes das regras (RHS) são usados para gerar novos nós até que o objetivo é atingido. No raciocinio para-tras, os lados direitos são comparados com o nó corrente e os lados esquerdos são usados para gerar novos nós representando novos estados objetivos a serem atingidos. O processo continua até que um destes estados objetivos seja instanciado por um estado inicial. Para decidir qual a direção a ser usada na busca devemos, por exemplo, considerar o que seria preferivel, se mover do menor conjunto de estados para o maior conjunto de estados ou utilizar a direção com menor fator de ramificação. Proceder, em caso do programa ser questionado sobre o raciocinio usado, na direção que corresponde mais aproximadamente à forma como o usuario pensa.

4.1.3 Decomposição do Problema.

Se estivermos de posse de conhecimento que nos indique ser possivel dividir o problema em problemas intermediários, devemos fazer uso desta informação, porque isto geralmente levará a um aumento da eficiência computacional. Um Sistema de Produção que permite este tipo de sub-divisão é chamado de Decomponivel [NILSSON (1980)].

Durante a busca da solução de um problema podem ser gerados caminhos redundantes e isso pode ser ineficiente porque a estratégia de controle pode tentar

explorar todos eles. Uma maneira de evitar a exploração de caminhos redundantes é reconhecer se a base de dados inicial pode ser decomposta em componentes que possam ser processados independentemente. Para decompor a base de dados, o sistema deve também estar apto a decompor a condição de parada. Isto é, se trabalharmos em cada componente separadamente, devemos estar em condições de expressar a condição de parada global usando a condição de parada de cada um dos seus componentes.

4.1.4 Busca Heurística.

Na ausencia de métodos formais, bem estruturados e eficientes, de solução para alguns tipos de problemas, o uso de heurísticas revela-se de grande utilidade e muitas vezes se constitui na única alternativa de solução. Uma heurística é uma técnica que aperfeiçoa a eficiência do processo de busca, possivelmente sacrificando requisitos de completude e otimalidade.

Em muitos problemas, nem a completude nem a otimalidade são essencias. Em vez disso, estamos interessados em descobrir heurísticas que possam levar a uma boa solução, no tempo disponivel. Costuma-se dizer das heurísticas, que estas obtem boas soluções na maior parte dos casos mas podem falhar estrondosamente em outros. Há algumas boas heurísticas de propósito geral. Por outro lado pode-se construir heurísticas especializadas para explorar o conhecimento de um domínio específico para resolver problemas particulares. O uso de heurísticas como o algoritmo de "seleção do vizinho mais próximo" ajuda, no caso do problema do caixeiro viajante, a evitar a explosão

combinatória. Ainda com relação à otimalidade, SIMON (1983) afirma que há evidências de que as pessoas em geral são "satisfazedoras" e não otimizadoras. Embora as aproximações produzidas por heurísticas possam não ser muito boas no pior caso, os piores casos raramente surgem no mundo real. Em geral se diz que uma heurística é "melhor informada" que outra na medida que incorpora mais conhecimento sobre o ambiente do problema, e o utiliza na busca da solução. Uma heurística "melhor informada" que outra ajuda a aumentar a eficiência da busca em relação à que seria conseguida pela segunda.

Informações do tipo heurístico podem ser utilizadas para ordenar os nós atingidos durante a busca em um grafo, de tal maneira que a busca se expanda ao longo dos nós terminais mais promissores. Uma forma de registrar a "promissividade" de um nó é através do calculo de funções de avaliação. Os critérios para definição de avaliação tem sido variados, tais como a probabilidade do nó estar no melhor caminho, métrica da distância do nó que está sendo avaliado e um nó da configuração objetivo, entre outros critérios. O mais difundido dos algoritmos que utilizam funções de avaliação é o A*. O A* utiliza a seguinte função de avaliação f :

$$f(n) = g(n) + h(n)$$

onde g(n) é o custo acumulado até o nó n a partir do nó inicial; h(n) é a estimativa do custo de atingir um nó da configuração objetivo a partir de n; se tal estimativa for otimista, pode-se mostrar que A* é "admissível", isto é, sempre termina com o caminho ótimo a partir do nó inicial S

até o nó objetivo caso tal caminho exista. A precisão da função heurística h depende da quantidade de conhecimento heurístico que esta possui sobre o domínio do problema. Um valor h(n) = 0 reflete a ausência de conhecimento heurístico aplicável na busca.

Podemos dizer que uma versão A_2 do A* é mais informada que outra A_1 se para nós não objetivos n, $h_2(n) > h_1(n)$. Podemos enunciar este fato da seguinte maneira :

"Se A_1 e A_2 são duas versões do A* tal que A_2 é mais informado que A_1, então ao término de suas buscas em qualquer grafo tendo um caminho de S até o nó objetivo, todo nó expandido por A_2 também é expandido por A_1. Segue-se que A_1 expande pelo menos tantos nós quando A_2".

O algoritmo de busca do caminho do método de Planejamento de Rotas descrito neste trabalho é uma versão especializada do A*.

4.2 Representação do Conhecimento.

Os métodos de Resolução de Problemas descritos na seção anterior são muito gerais e ineficientes na resolução de problemas complexos do mundo real. Foi percebido pelos pesquisadores de I.A. que para a resolução destes últimos problemas seria mais eficiente acumular uma grande quantidade de conhecimento sobre o problema específico na base de dados global de que tentar construir poderosos mecanismos gerais de controle. Apesar de estes métodos serem úteis e formarem o arcabouço de muitos métodos de solução que utilizam representação de conhecimento, seu valor é limitado por causa de sua generalidade.

4.2.1 Lógica de Predicados.

Quando o conhecimento a ser representado na base de conhecimento consiste em declarações descritivas independentes do seu grau de verdade ou precisão, uma forma de representação que surge naturalmente é através do uso da lógica de predicados de primeira ordem. Esta é uma linguagem formal em que uma ampla variedade de declarações pode ser representada.

O uso do formalismo lógico é interessante porque imediatamente sugere uma maneira poderosa de derivar novo conhecimento a partir do antigo : dedução matemática. Neste formalismo podemos concluir que uma nova afirmação é verdadeira pela prova de que é consequência das afirmações ou declarações que já são conhecidas. Uma expressão definida corretamente em lógica de predicados é denominada de wff ("well-formed-formula"). A seguir são mostrados alguns exemplos de representação de declarações através da lógica de predicados :

1) Oswaldo é pediatra

Pediatra(Oswaldo)

2) Todos os pediatras são médicos

Para todo x, Pediatra(x) ==> médico(x)

3) Os médicos que trabalham no Hospital Getulio Vargas são residentes ou contratados

Para todo x, Trabalha(x,Getulio-Vargas) ^ medico(x) ==>
residente(x,Getulio-Vargas) v
contratado(x,Getulio-Vargas)

onde Pediatra(Oswaldo), médico(x) são exemplos de fórmulas chamadas atômicas.

4.2.1.1 Unificação.

Na prova de teoremas envolvendo formulas quantificadas, é muitas vezes necessário comparar certas sub-expressões. A finalidade da comparação é descobrir se existe alguma substituição que forme um par ou pares de sub-expressões idênticos. Existe um procedimento recursivo direto para fazer isto denominado Unificação. A descrição do algoritmo de unificação UNIFICA é dada em RICH (1986). O algoritmo recebe como argumentos os literais L1 e L2 e retorna como o seu valor uma lista representando a composição das substituições que foram executadas durante o processo de "matching" (comparação). O retorno da lista vazia NIL indica que foi detectado um "casamento" de padrões sem efetuar qualquer substituição. Por outro lado, se a lista consiste do valor indicativo de falsidade F, é indicação de que o processo de unificação falhou.

4.2.1.2 Resolução em Lógica de Predicados.

A Resolução é uma importante regra de inferência que pode ser aplicada a certa classe de wffs chamadas cláusulas. Uma cláusula é definida como sendo uma wff consistindo de uma disjunção de literais (literais são fórmulas atômicas ou a negação de fórmulas atômicas). Previamente à descrição do processo de Resolução, daremos algumas definições importantes extraidas de PINHO (1986). Tautologias da forma P => Q são bastante usadas em sistemas

formais de dedução (P é chamado de antecedente tautológico e Q, consequente tautológico). Chamaremos de implicação tautológica a tautologia P => Q. Estas tautologias estão diretamente relacionadas com as chamadas regras de inferência, que permitem obter wffs verdadeiras a partir de um conjunto de wffs dadas (estas consideradas verdadeiras). Por exemplo, a regra "Modus Ponens" permite obter a fórmula Q a partir do conjunto {P, P => Q }. Note que (P ^ P => Q) => Q é uma tautologia. Como P => Q é equivalente a ~P v Q, Modus Ponens pode ser expresso por :

(a) [P ^ (~P v Q)] => Q

Outras formas equivalentes para Modus Ponens são por exemplo as seguintes :

(b) [~P ^ (P v Q)] => Q

(c) [(P v Q_1 v Q_2 v...v Q_N) ^ ~P] => Q_1 v Q_2 v...Q_N

(b) foi obtida de (a) substituindo-se P por ~P e ~~P por P; (c) foi obtida de (a) trocando-se a posição de P e (~P v Q), substituindo-se Q por Q_1 v Q_2 v...Q_N, P por ~P e ~~P por P.

Chama-se Resolução o processo de obter novas wffs a partir das anteriores, e esse processo pode ser representado por um grafo, chamado de grafo de derivação. As wffs obtidas por regras de inferência são chamadas teoremas, e a forma de obtê-las, prova de teorema; no caso de

resolução, as wffs utilizadas são chamadas "pais", e a wff obtida, "resolvente".

O processo de resolução pode ser aplicado iterativamente, partindo-se de um conjunto de wffs, indicando-se cada nova wff obtida no conjunto original. Para que o processo de resolução possa ser realizado, é necessário que cada wff esteja na forma de cláusula. A descrição do procedimento para realizar esta transformação pode ser achada em PINHO (1986). Normalmente estamos interessados em provar uma determinada afirmação; e uma forma de fazê-lo é negar a resposta que queremos obter, incluí-la como uma das cláusulas dadas, e, por resolução, procurar obter NIL. Este processo de Resolução é conhecido por Refutação. A seguir daremos um exemplo para ilustrar o processo de Resolução por Refutação.

Sejam as seguintes expressões do Cálculo de Predicados :

1. ALTO(JOAO) v ~INTELIGENTE(JOAO)
   "Ou João é alto ou não é inteligente".

2. ALTO(MARIO) v INTELIGENTE(JOAO) v GORDO(MARIO)
   "Ou Mario é alto ou João é inteligente ou Mario é gordo".

3. ~GORDO(MARIO) v ALTO(JOAO)
   "Ou Mario não é gordo ou João é alto"

4. ~ALTO(MARIO)
   "Mario não é alto"

Desejamos provar que João é alto, então acrescentamos a negação desta afirmação :

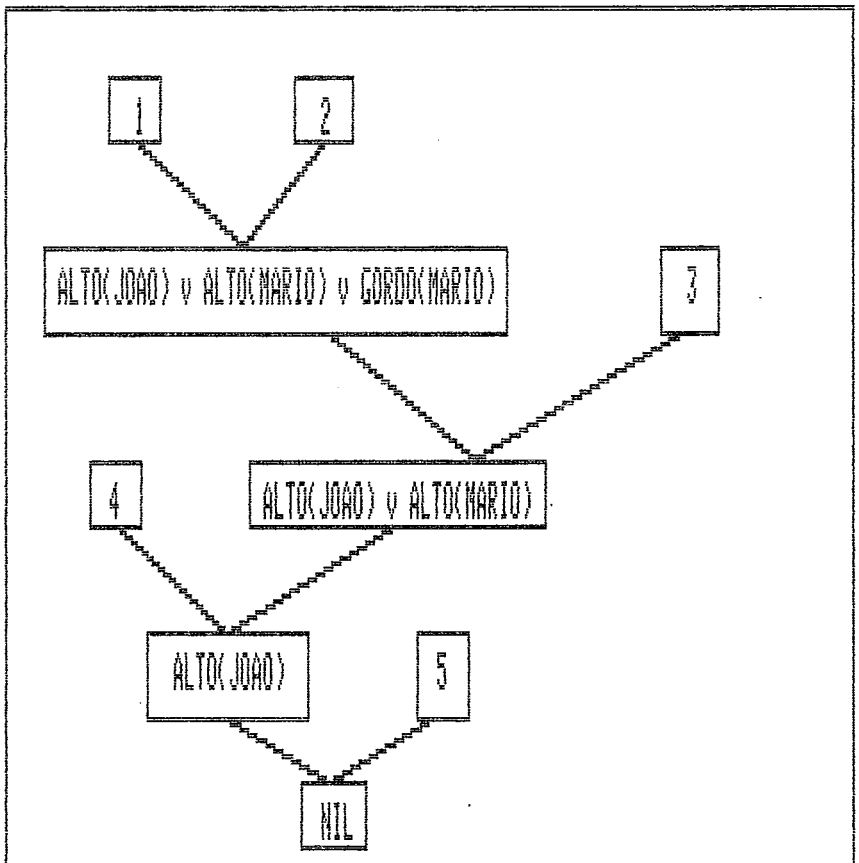


FIGURA IV.2 - GRAFO DE DERIVACAO DA PROVA DE 5 POR REFUTACAO

5. ~ALTO(JOAO)

O grafo de derivação da prova por Refutação para este exemplo encontra-se na figura IV.2.

O processo de Refutação consiste na tentativa de verificar se P => Q é uma implicação tautológica, sendo P a conjunção das fórmulas ocorrendo na base de dados e Q a tese que desejamos provar. Como P => Q é equivalente a ~P v Q, verificar se P => Q é uma tautologia equivale a verificar se P ^ ~Q (que equivale a ~(~P v Q) é uma contradição. Isto é se P ^ ~Q é equivalente a NIL.

Em Cálculo de Predicados não existe um procedimento que decida a contradição (ou a tautologia) de uma wff, por isso o Cálculo de Predicados é dito indecidível. Ou seja, se o sistema for contraditório, é possível ter um algoritmo que, eventualmente, encontre NIL, no processo de Refutação; mas, se o sistema não for contraditório, tal algoritmo pode nunca terminar. A definição do algoritmo geral de Refutação pode ser achada em PINHO (1986).

4.2.1.3 Lógica de Predicados em I.A.

Na Seção 4.2.1.1 vimos o processo de unificação que tipicamente é usado para descobrir se um literal pode "instanciar" outro. Pode haver variáveis em ambos literais, e estas variaveis podem ter seus termos substituidos por aqueles que fariam os literais idênticos. O processo de comparação entre uma expressão e uma outra expressão padrão é algumas vezes denominada de "pattern-matching" (comparação de padrões). O processo de unificação é mais geral que o que comumente se chama de

"pattern-matching", porque neste último não são permitidas as ocorrências de variáveis em ambas expressões.

Podemos usar Sistemas de Produção para tentar mostrar que uma dada formula, chamada de wff objetivo, é um teorema derivável de um conjunto de formulas (descrições de estado). Os Sistemas de Produção deste tipo são denominados de Sistemas de Provas de Teoremas ou Sistemas de Dedução. Os Sistemas de Provas de teoremas tem aplicação imediata em Matemática e Lógica. Uma outra aplicação importante é nos Sistemas de recuperação inteligente de informação, onde deduções são feitas a partir de uma base de dados de fatos para derivar uma resposta para uma pergunta. Por exemplo consideremos as seguintes wffs :

GERENTE(DEPTO-COMPRAS,JOAO-PAULO),
"João Paulo é gerente do Departamento de Compras"

TRABALHA-EM(DEPTO-COMPRAS,CARLOS-ALBERTO),
"Carlos Alberto trabalha no Departamento de compras"

e

[TRABALHA-EM(x,y) ^ GERENTE(x,z)] ==> CHEFE-DE(y,z)
"Se y trabalha na loja x e z é gerente da loja x então z é chefe de y"

A partir destas informações um Sistema deste tipo pode responder à pergunta "Qual é o chefe de Carlos Alberto?". Esta pergunta pode ser formulada como um teorema a ser provado :

(Existe um x), CHEFE-DE(CARLOS-ALBERTO,x)

Uma prova construtiva (ou seja aquela que mostra o valor de

x) providenciaria uma resposta para a pergunta.

Muito do conhecimento utilizado pelos Sistemas de I.A. é diretamente representável por expressões implicacionais genéricas. Os Sistemas de Dedução baseado em regras não convertem wffs para cláusulas. As wffs são utilizadas em uma forma próxima de sua forma original. As wffs representando o conhecimento sobre o problema são separadas em duas categorias : regras e fatos.

Os fatos são as afirmações que não são expressas como implicações, eles servem para representar conhecimento especifico sobre um determinado assunto. Uma prova de uma wff objetivo pode ser obtida a partir de fatos e regras. Muitos dos denominados Sistemas Especialistas utilizam esta técnica de representação e utilização do conhecimento. Este formato situação-ação é particularmente adequado para expressar o conhecimento que muitos especialistas em algum campo têm, como resultado da experiência. A representação do conhecimento por meio de regras tem como desvantagens a falta de variação e de estruturação. Seu formato é inadequado ou inconveniente para representar muitos tipos de conhecimento tal como a compreensão de conhecimento causal ou a modelagem da estrutura de um Sistema. Sua falta de variabilidade na expressão de conhecimento também limita a representação de conhecimento causal. O conhecimento causal é o comumente requerido pelo ser humano quando suas regras disponíveis falham na tentativa de obter uma resposta satisfatória.

A medida que o número de regras em um Sistema cresce, fica dificil sua manipulação e modificação. Apesar de que os Sistemas baseados em regras são muito modulares,

as regras não são tão independentes como aparentam à primeira vista. Elas são independentes apenas no sentido em que não são estruturadas e encontram-se frequentemente fora de sequência. Porém assumem informações ou ações obtidas em outras regras, e trabalham em conjunção com estas. O acréscimo da base de conhecimento não pode então ser realizado de forma indefinida, pois conforme o número de regras cresce, serão estabelecidos pressupostos e relações implícitas entre as regras. Como resultado, fica fácil adicionar uma nova regra que viole algum pressuposto ou relação previamente estabelecidos. Se acontece alguma violação no sistema baseado em regras, o sistema não degrada paulatinamente e pode seguir uma via errada de raciocínio, sendo possível o retorno de respostas completamente inesperadas.

Um outro elemento dos Sistemas baseados em conhecimento é o Motor de Inferência ou Interpretador, cuja função é decidir qual regra será ativada a seguir. Uma descrição do funcionamento de um Interpretador simples é feita a seguir :

I) Ache todas as regras cujo LHS seja verdadeiro e marque-as como aplicáveis.

II) Se mais de uma regra é aplicável, então desative qualquer regra cujo RHS adicione um símbolo duplicado para a lista CONTEXTO (fatos).

III) Execute a ação da regra aplicável de menor ordem. Se nenhuma regra é aplicável então finalize o processo.

IV) Restabeleça a aplicabilidade de todas as regras e retorne para o passo I.

A performance deste Intepretador é ineficiente e esta será analisada mais detalhadamente no Capitulo VI nesse capitulo também será descrito um interpretador mais eficiente.

4.2.2 Frames.

MINSKY (1975), propôs a teoria de frames, um mecanismo de representação de conhecimento em computadores. Um frame é uma estrutura que representa conhecimento de um dominio muito limitado. Um frame produz uma descrição do objeto ou ação em questão começando com uma estrutura comum invariavel para todos os casos do seu dominio e adicionando certos aspectos de acordo com observações particulares. A descrição resultante é estabelecida em termos de um número limitado de descritores. Algumas propiedades importantes dos frames como uma representação para o conhecimento são listadas a seguir :

Descrição : O frame provê uma estrutura elaborada para criar e manter uma descrição. Um elemento primitivo desta descrição pode ser expandido para um frame quando sua descrição interna é de interesse.

Instanciação : O frame produz uma descrição do objeto sendo examinado pela substituição dos valores observados por valores preditos.
Na falta de valores observados são assumidos valores "default".

Predição : A descrição predita do frame pode ser usada para guiar a coleção de observações para

instanciação. Produz tambem os valores "default".

Justificação : Há diversos graus de confiabilidade nos aspectos que fazem parte da descrição, alguns são resultado de observações diretas, outro de escolha entre alternativas e alguns são valores "default".

Variação : As dimensões e faixas de variação dos diversos aspectos são limitados e estabelecidos.

Correção : Ocorrencias de anomalias podem indicar que o frame corrente não é correto e que é necessario um ponto de vista diferente. O frame analisa a anomalia e se for necessario realiza uma substituição mais apropriada.

Perturbação : Para pequenas mudanças no observador ou no observado, procedimentos de tratamento de perturbações corrigem a descrição sem precisar de uma completa recomputação.

Transformação: No caso de ocorrencias mais significativas os procedimentos de transformação propõem frames mais adequados para a nova situação.

Devido ao fato de que o conhecimento interrelacionado é agrupado, os frames e os Sistemas baseados em frames que os contêm, estruturam a informação em uma forma muito mais organizada e manipulável que os Sistemas baseados em regras. Os frames podem ser altamente

complexos e contêm em adição aos dados, uma variedade de tipos de informação, conhecimento causal, e relações, que as bases de dados não podem manipular. Os frames também podem ser ligados a outros frames e podem "herdar" informações deles, isto porque em um Sistema baseado em frames, os frames podem conter outros frames menores, chamados de sub-frames, organizados em uma hierarquia. Assim, por exemplo, em um frame, podemos possuir nos slots, descrições da entidade a qual o frame se refere, valores de atributos, ponteiros para outros frames subordinados ou para um frame "pai", nomes de procedimentos a serem executados ou também regras de produção.

Ao contrário dos Sistemas baseados exclusivamente em regras de produção, a degradação do Sistema, á medida que atinge o limiar do conhecimento é gradual.

4.2.3 Representação do Conhecimento e o "Frame Problem".

Como objetos e fatos podem ser representados ? Como a representação de objetos individuais pode ser combinada para formar a representação de um estado do problema completo ?. A resposta para estas perguntas constituem o problema de representação do conhecimento. Algumas tecnicas de representação já foram abordadas e são a lôgica de predicados e as técnicas estruturadas de repreSentaçãoo do conhecimento (frames). Estas últimas baseiam-se na ideia de que o conhecimento pode ser representado como um conjunto de objetos, cada um com uma coleção de atributos e um conjunto de relações com os outros objetos.

A questão de representar eficientemente as sequencias de estados de um problema que surgem no processo de busca de uma solução, é conhecida como o "frame problem". Em alguns dominios, a parte mais dificil é a representaçao de todos os fatos. Em outros, especificar o quê muda, não é trivial. A melhor abordagem para tratar com este tipo de problema depende do tipo de estados do mundo e ações que estamos modelando. Exemplificando, no mundo simples do robô, se os componentes de um estado do mundo estão muito proximamente acoplados ou são instáveis, então cada ação pode ter um efeito profundo nos efeitos globais no estado do mundo. Em tal mundo, pegar um bloco de uma pilha, pode derrubar a pilha inteira. Tipicamente, os componentes de um estado do mundo são suficientemente desacoplados para permitir que os efeitos de uma ação sejam relativamente locais.

O "frame problem" geralmente é mais critico para modelar detalhes porque devem levar em consideração acoplamentos entre componentes do estado do mundo que poderiam ser ignorados em niveis elevados de planejamento.

4.2.4 Representação do Conhecimento no Problema de Planejamento de Rotas.

Nas seções 4.2.1 e 4.2.2 foram abordadas duas importantes formas de representação do conhecimento. Existem muitas outras representações tais como Redes Semânticas, Scripts, etc. Uma exposição de cada uma destas representações é desnecessária uma vez que é feita na literatura especializada. As únicas representações que aqui nos interessam foram as descritas neste capitulo

porque são as utilizadas no método de planejamento de rotas descrito no trabalho.

Muito do conhecimento que o projetista possui sobre uma configuração problema é do tipo heurístico e pode ser expressado naturalmente através de declarações (fatos) e regras usando lógica de predicados. A arquitetura de um Sistema baseado em conhecimento pode ser utilizada aqui também para representação do conhecimento.

Através de Inferência é possivel derivar outras informações. Por exemplo, o axioma

"Para todo X, (X é caldeira) c (X é quente)" afirmando que todas as caldeiras são quentes, e a declaração :

O1 é caldeira

pode ser representada pela regra

A,

A está contido em B ==> B

que conduz á

O1 é quente

Este tipo de representação nos permite a reduçäo do conhecimento de um nivel mais alto para conhecimento do tipo primitivo tal como foi visto no Capitulo III. Por outro lado o problema de planejamento de rotas no escopo definido neste trabalho possui um dominio reduzido o que nos leva a existência de regras de pouca variedade e em número não muito grande. Estas caracterisiticas ajudarão a manter a consistência de nossas regras, o que se torna dificil em Sistemas com base de

conhecimentos com número elevado e variado de regras.

O Módulo de Inferência desenvolvido para o Método garantirá a eficiencia do processo de dedução. O conhecimento sobre o problema é armazenado em uma base de conhecimento; nela encontram-se armazenadas a descrição de restrições, índices de performance de complexidade variável, assim como heurísticas que representam estratégias para uma busca mais eficiente da rota ótima. Este conhecimento, como foi mencionado anteriormente encontra-se parcialmente sob a forma de fatos e regras de produção, uma forma de representação conveniente para conhecimento do tipo heurístico. Este conhecimento não se encontra necessàriamente agrupado na base de conhecimento, mas uma vez selecionado se faz necessário uma representação deste conhecimento que permita sua utilização eficiente e direta pelo módulo de busca do caminho. A representação do conhecimento nesta última etapa é feita por meio de frames que apresentam as seguintes vantàgens :

- O conhecimento necessàrio na determinação da rota ótima é distribuido em "slots" no frame, possibilitando assim a agrupação de diversos tipos de informação que se encontram inter-relacionados, assim como sua localização imediata quando necessàrio.
- A representação por frames permite o mecanismo de "hereditariedade" com o uso de sub-frames ou frames "filhos". Este mecanismo consiste na transmissão ou "herança" de conhecimento entre os frames de maior hierarquia e os de menor hierarquia ou no sentido inverso quando necessàrio. Este mecanismo nos permitirà a

decomposição da rota quando possivel, "herdando" cada um dos sub-frames as informações necessárias do frame "pai" e retornando os resultados obtidos para este.

A utilização de frames é recomendável quando o conhecimento englobado é interrelacionado e o assunto do conhecimento é limitado, que é o caso do problema de planejamento de rotas descrito neste trabalho.

4.3 Linguagens de I.A.

Existem aspectos específicos importantes em uma linguagem para escrever programas de I.A. distintos dos de uma linguagem para escrever outros tipos de programas :

-Boas facilidades para manipular listas, posto que as listas são uma estrutura amplamente usada na maioria dos programas de I.A.
-facilidades de "pattern-matching".
-facilidades para executar alguns tipos de dedução automática.
-facilidades para construir estruturas de conhecimento complexas.
-mecanismos de interação com o programador.
-estruturas de controle que facilitem o comportamento dirigido para um objetivo (top-down), conjuntamente com o comportamento convencional dirigido para os dados (bottom-up).
-habilidade de mesclar procedimentos e estruturas de dados declarativos de maneira que se adapte para uma tarefa específica.

Nenhuma linguagem existente possui todas

estas características; na maior parte dos casos elas possuem mais habilidade em alguns itens, em detrimento de outros.

4.3.1 LISP.

A linguagem LISP foi criada em 1958 por John Mc Carthy e é ainda hoje a linguagem mais utilizada em I.A. As principais ideias de LISP, descritas por Mac Carthy, são:

1. Computação com expressões simbólicas ao invés de números; isto é, poucos padrões na memória e registros do computador podem acumular-se para formar símbolos arbitrários, não somente aqueles da aritmética.
2. Processamento de listas, isto é, representando dados como estruturas de listas encadeadas na máquina e como listas de multi-níveis no papel.
3. Estrutura de controle baseada na composição de funções para formar funções mais complexas.
4. Recursão como uma maneira de descrever processos e problemas.
5. Representação dos programas em LISP internamente como listas encadeadas e externamente como listas multi-níveis, isto é, na mesma forma em que os dados são representados.
6. A função EVAL , escrita em LISP serve como um interpretador para o LISP e como uma definição formal da linguagem.

Em LISP não há então diferença formal entre dados e programas, isto é, programas em LISP podem usar outros programas em LISP como dados. A linguagem LISP é altamente recursiva, e tanto programas como dados são

representados como listas.

Existem poucas funções básicas em LISP; todas as outras funções são definidas em termos destas funções básicas; pode-se construir fàcilmente novas funções de alto nível.

4.3.2 OPS5.

OPS5 e as versões anteriores da familia OPS tem sido usadas para uma variedade de aplicações nas áreas de I.A. e psicologia cognitiva. Uma das primeiras aplicações de OPS na área de Engenharia de conhecimento foi o sistema R1 para configuração de computadores VAX.

O OPS5 incorpora mecanismos de controle e representação gerais. Apesar de prover os mecanismos básicos necessários para engenharia do conhecimento, o OPS5 não está orientado para estrategias particulares de "problem-solving" ou esquemas representacionais.

O OPS5 permite ao programador usar símbolos e representar relações entre os símbolos, mas os símbolos ou relações não tem significados pre-definidos. Os significados são determinados completamente pelas regras de produção escritas pelo programador.

O mecanismo de controle do interpretador OPS5 é um "loop" simples chamado de ciclo "reconhece-atua", que o usuario elabora como deseja. Ao invés de mudar o interpretador, o programador escreve regras para efetuar as estratégias de controle escolhidas.

4.3.3 PROLOG.

A linguagem PROLOG (Programação em Lógica), foi desenvolvida inicialmente em 1972 por A. Colmerauer e F.

Roussel na Universidade de Marseilles. O PROLOG é uma linguagem que implementa uma versão simplificada do cálculo de predicados e é desta forma uma linguagem lógica autêntica.

Como o LISP, o PROLOG esta projetado para computação simbólica. O PROLOG é uma linguagem interpretada e responde a qualquer pergunta tentando retornar uma resposta imediatamente.

Para programar em PROLOG o procedimento é o seguinte :

1. Especificar alguns fatos sobre objetos e relações.
2. Especificar regras sobre objetos e relações.
3. Responder perguntas sobre objetos e relações.

Assim, se for dada entrada aos seguintes fatos :

Gosta(João,Maria)

Gosta(Paulo,Maria)

Gosta(Maria,João)

e perguntar : ? Gosta(João,Maria)

o PROLOG responderá :

SIM

O PROLOG é a melhor implementação de programação lógica, apesar de não poder manipular todas as deduções que são teoricamente possíveis no cálculo de predicados. Ao mesmo tempo, a sintaxe do PROLOG é muito menos complexa que a maioria das linguàgens de programação convencionais de poder semelhante.

Existem dois estilos de programação em PROLOG : O estilo declarativo e o estilo procedural. Na programação

declarativa, a preocupação é em passar ao sistema o que este deve conhecer e deixar com o sistema a manipulação de procedimentos. Na programação procedural, consideramos o comportamento específico de resolução de problemas que o sistema exibirá. A abordâgem declarativa é a que tem feito de PROLOG uma linguâgem muito popular, pois o trabalho se restringe nesse caso em passar ao sistema o que deve ser feito, ficando uma tarefa da maquina o "como" faze-lo.

CAPITULO V - MODULO DE INFERENCIA

5.1 Aspectos principais do Módulo de Inferência.

Na descrição do método de Planejamento de Rotas vimos que o módulo de inferência desempenha um papel de extrema importância. Através dele é que faremos a redução do conhecimento fornecido em um nível elevado para conhecimento do tipo primitivo, será ele que possibilitará também a verificação pelo sub-módulo de avaliação do módulo de busca do caminho se uma determinada cela possui algum ou vários tipos de restrições e dos parâmetros envolvidos, informações estas que serão utilizadas pelas funções que compõem o sub-módulo de avaliação que verificará se estas são satisfeitas pela cela atingida. A implementação do método possue então características de Sistemas de Produção, no qual a representação do conhecimento é feita através de regras de produção, frames e celas obstáculo-adaptativas.

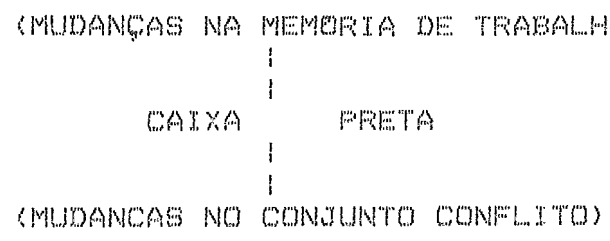
Um Modulo de Inferência convencional costuma ser ineficiente porque executa muitas operações desnecessárias. Primeiramente, uma mesma cláusula teste pode aparecer em multiplas regras, e nós somos obrigados a confrontar repetidamente a mesma cláusula teste com as cláusulas da memória de trabalho (working-memory). Em segundo lugar, em cada ciclo de "matching" nós começamos de novo a confrontar nossas regras contra todas as cláusulas na memória de trabalho. Quando nós entramos em um novo ciclo, porém todas as cláusulas da memória de trabalho do ciclo anterior são re-carregadas, apesar de somente uma nova cláusula ter sido adicionada. Desta forma a maior parte das cláusulas de memória de trabalho sobre as quais estamos

executando o processo de "matching", já passaram por este processo no ciclo anterior. Por outro lado, quando confrontamos uma regra de produção contra a memória de trabalho, um bom esforço de computação é requerido para fazer o "merge" de cláusulas de teste e construir ações (RHS - Right Hand Side da regra). Cada vez que fazemos o "matching" de uma regra de produção , esta computação é repetida. É mais eficiente fazer com que nosso programa examine as regras antes de executar o "matching", determinar quais as variáveis que se repetem no LHS (Left Hand Side) da regras e quais aparecem no RHS desta. Então o programa pode construir um segmento de codigo em LISP para cada regra de produção que irá fazer diretamente o "merge" das cláusulas e construir ações.

Neste capitulo é descrita uma implementação alternativa para módulos de inferência que incorpora caracteristicas do módulo de inferência RETE descrito em FORGY (1982) e do módulo descrito em ANDERSON (1987). O reconhecedor de padrões evita interagir sobre os elementos na memória de trabalho, armazenando informação en cada ciclo. A etapa que pode requerer iteração é a determinação de se um determinado padrão de regra coincide com o de alguns dos elementos da memória de trabalho. Os interpretadores mais simples (e ineficientes) determinam isto pela comparação do padrão com os elementos um por um. A iteração pode ser evitada pelo armazenamento em cada "padrão" (predicado), da lista de elementos que conferem e lhe estão associados. Estas listas são atualizadas quando ocorrem mudanças na memória de trabalho. Quando uma cláusula é adicionada para a memória de trabalho, o interpretador de

regras faz a adição da lista com os elementos associados ao predicado desta cláusula, à multilista contendo as listas de atributos anteriormente incluidas. Desta forma nunca é necessário examinar a memória de trabalho, chamada em nossa implementação de CONTEXTO.

O reconhecedor de padrões do módulo de inferência pode ser visto como uma caixa preta com uma entrada e uma saida.



O conjunto conflito é aquele que contém a informação das cláusulas geradas por instanciações feitas nas regras de produção mais o rótulo de identificação destas regras e que não foram ainda selecionadas e portanto não adicionaram ainda elementos para a memória de trabalho. O conjunto conflito pode conter mais de uma instanciação para uma determinada regra. A Caixa Preta recebe a informação das mudanças que são feitas na memória de trabalho, e determina as mudanças que devem ser feitas no conjunto conflito para mantê-lo consistente, como por exemplo remoção dos elementos repetidos e critérios para seleção da regra a ser ativada.

A estrutura das regras a serem aqui utilizadas é a seguinte:

(rótulo-da-regra((Condição 1)(Condição 2)...(Condição N))
==> (Ação))

Exemplo :

(P1 ((QUENTE =OT)(OBSTACULO =OT)(SENSIVEL-AO-CALOR =RX)) ==>
(DISTANCIA-MINIMA =RX =OT 5))

A regra possue o rôtulo de identificação P1 e significa :

"Se um obstáculo é quente e a rota é sensivel ao calor então a distância mínima de obstáculo para a rota é igual a 5"

Os elementos na regra precedidos do símbolo "=" são variáveis. As variáveis são usadas para dar generalidade às regras.

A estrutura dos elementos do conjunto conflito é a seguinte :

(lbl (clause))

onde: lbl = rôtulo da regra de produção cujo LHS foi instanceado.

clause = cláusula gerada pela ativação do RHS desta regra.

Exemplo : (P1 (DISTANCIA-MINIMA R1 O3 5))

P1 -> Indica que a cláusula foi gerada por ativação do RHS da regra P1.

(DISTANCIA-MINIMA R1 O3 5) -> representação simbólica de cláusula significando : "A distância mínima da rota R1 para o obstáculo O3 é igual a 5.

O Interpretador de regras do módulo de

inferência não começa de novo a cada ciclo, confrontando as regras de produção contra todas as cláusulas da memória de trabalho. Além da adição de elementos na lista associada ao atributo de um novo elemento descrita anteriormente, o interpretador de regras verifica se o novo elemento em conjunto com as cláusulas adicionadas anteriormente possibilitam a instanciação de algumas regras da lista REGRAS (A relação de regras de produção do sistema encontra-se na lista REGRAS). Devido ao esquema usado, não é necessário, em nenhum instante do processamento do módulo de inferência, a interação com os elementos da memória de trabalho ("CONTEXTO"). Se é possível a instanciação, os elementos contendo o rótulo da regra de produção conjuntamente com a cláusula produto da execução do RHS da regra são adicionadas ao conjunto conflito.

O processamento de novas cláusulas é feito por uma função denominada em nossa implementação de ENTRA-CLAUSULA. No primeiro ciclo, todas as cláusulas iniciais da memória de trabalho passarão pelo processo de entrada. Em cada ciclo sucessivo, somente uma nova cláusula é entrada, aquela que foi adicionada à memória de trabalho no ciclo anterior (Resultado da ativação de uma instanciação de uma regra do conjunto conflito). Assim, depois que todas as novas cláusulas foram entradas em um ciclo, uma regra do conjunto conflito é ativada, e as instanciações remanescentes no conjunto conflito são salvas. Se o interpretador de regras não detecta novas cláusulas a serem adicionadas, ele seleciona uma do conjunto conflito e um novo ciclo é realizado. Desta maneira evitamos ter de gerar novamente cláusulas que foram geradas em ciclos anteriores

e que não foram selecionadas anteriormente pela função de resolução de conflitos.

A resolução de conflitos pode seguir estratégias complexas particulares segundo o caso. Em nossa implementação adotamos o critério simples de resolver o conflito de elementos candidatos selecionando sempre o primeiro elemento do conjunto conflito. No novo ciclo então, o interpretador de regras dá entrada somente a uma nova cláusula e acha todas as instanciações de regras que são possíveis a partir desta cláusula. As novas instanciações são adicionadas no conjunto conflito salvo no ciclo anterior, e uma nova instanciação no conjunto conflito é ativada e removida. Se não há mais instanciações no conjunto conflito depois de entrar uma nova cláusula no ciclo (ou depois de entrar todas as cláusulas no primeiro ciclo) então não há mais inferências que podem ser extraidas e a execução termina.

Nosso módulo de inferência realiza o processamento em duas etapas :

a) Codificação de regras.
b) Interpretação de regras.

Assim a função principal do módulo pode ser codificada :

```
(DEFUN MOTOR-DE-INFERENCIA ()
    (CODIFICA-REGRAS)
    (INTERPRETADOR-DE-REGRAS) )
```

A listagem com o código da implementação do módulo de inferência encontra-se em anexo. Cada cláusula pode ser codificada da seguinte maneira em LISP :

(PREDICADO TERMO-1 TERMO-2 ... TERMO-N)

Exemplo : (AREA-PROIBIDA R1 A1) para "A1 é área proibida para a rota R1"

5.2 Codificação de Regras.

Nesta etapa são executadas uma série de tarefas de forma a permitir um processamento mais eficiente do módulo de inferência. Estas tarefas consistem na criação de ponteiros de propriedades entre a regra e os predicados do LHS e na elaboração de codigo especializado para detectar casamento de padrões (propriedade "CONDICAO") e codigo para ativação do RHS (propriedade "ACAO"). Consideraremos como exemplo para descrição do mecanismo de codificação de regras a seguinte regra :

(P5 ((AREA =AX)(QUENTE =AX)(SENSIVEL-AO-CALOR =RX)) ==> (AREA-PROIBIDA =AX =RX))

que significa: "Se uma determinada área é quente e a rota é sensivel ao calor então esta área é uma área proibida para a rota"

A regra de produção P5 podemos associar várias propriedades, uma para cada cláusula teste com nome indicativo da posição relativa da cláusula no LHS e que tem como valor individual o predicado contido na cláusula :

(PUT P5 'PP1 'AREA) Atribue à regra P5 a propriedade PP1 com conteúdo igual a AREA.

(PUT P5 'PP2 'QUENTE) Atribue à regra P5 a propriedade PP2 com

conteúdo igual a QUENTE.

(PUT P5 'PP3 'SENSIVEL-AO-CALOR) Atribue à regra P5 a propriedade PP3 com conteúdo igual a SENSIVEL-AO-CALOR.

Por outro lado, podemos atribuir aos predicados do LHS, estes mesmos nomes de propriedades (PP1, PP2, PP3). O valor atribuido a cada uma destas propriedades é o resultado da atualização do valor anterior da propriedade pela inclusão do rótulo da regra onde o predicado aparece, que em nosso caso é a regra P5. O valor anterior da propriedade é a lista contendo os rótulos das regras onde anteriormente se detectou a presença do predicado na mesma posição relativa que em P5. Assim para a regra P5 temos o processamento das seguintes instruções :

(PUT 'AREA 'PP1 (CONS 'P5 (GET 'AREA 'PP1)))

-Coloca como valor da propriedade PP1 do predicado AREA, a lista resultante de adicionar o rótulo de regra P5 à lista obtida pela recuperação do conteúdo da propriedade PP1 deste predicado. O valor resultante desta operação poderia ser por exemplo a lista (P5 P4 P2), caso o predicado AREA apareça também na primeira cláusula teste das regras P4 e P2.

(PUT 'QUENTE 'PP2 (CONS 'P5 (GET 'AREA 'PP2)))

-Idem, para o predicado QUENTE que aparece na segunda cláusula teste da regra P5.

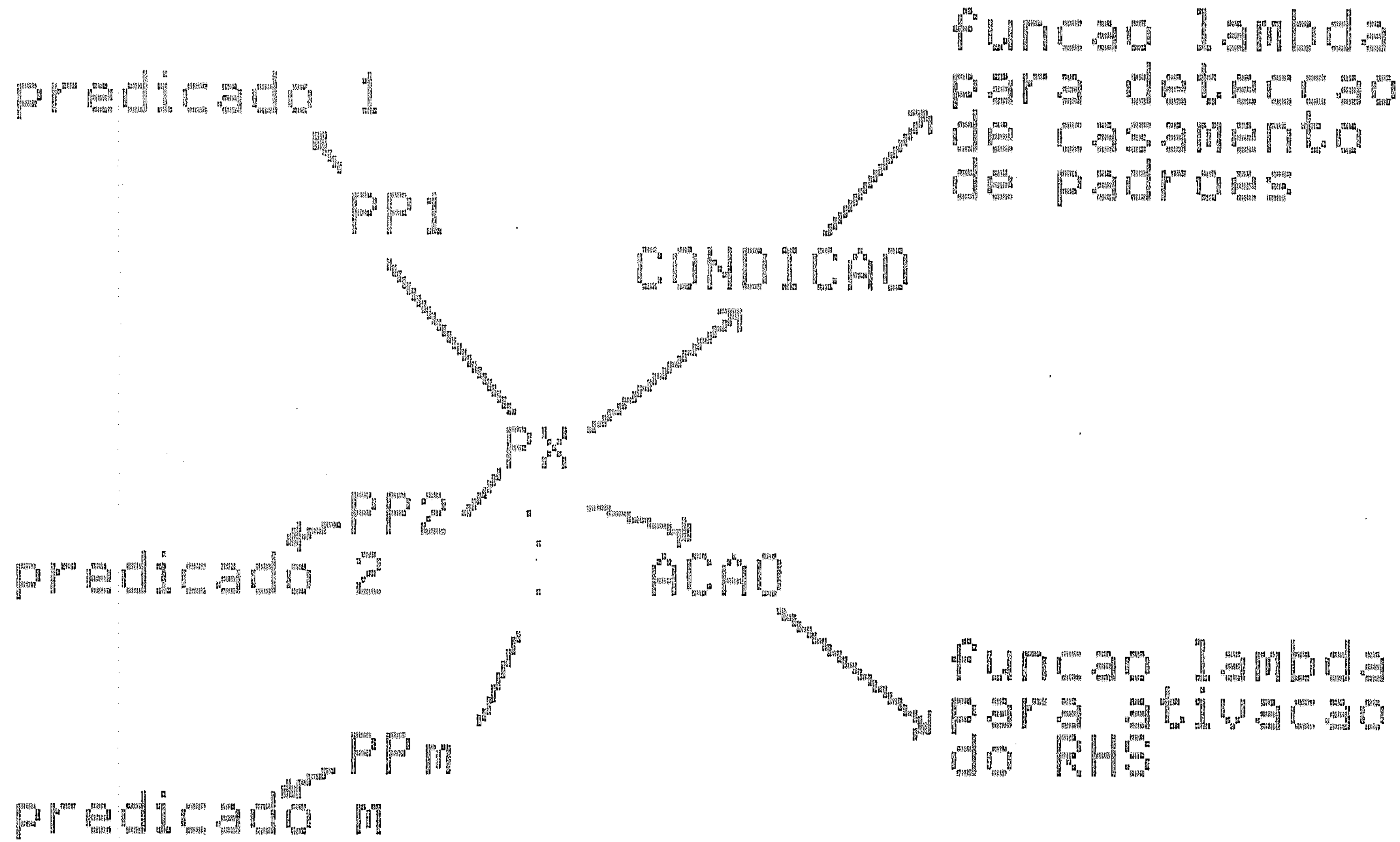
(PUT 'SENSIVEL-AO-CALOR 'PP3 (CONS 'P5 (GET 'AREA 'PP3)))

-Idem, para o predicado SENSIVEL-AO-CALOR que aparece na

terceira cláusula teste da regra P5.

Um predicado pode aparecer em cláusulas de posições relativas diferentes em regras diferentes do conjunto de regras do Sistema de Produção. Nesse caso, o predicado possuirá mais de uma propriedade associada. Desta maneira, o processo de codificação de regras possibilita processar as cláusulas da memória de trabalho mais eficientemente, pois podemos checar imediatamente para uma determinada cláusula, recuperando as propriedades do seu predicado (PP1, PP2, ...) quais as regras com alguma possibilidade de permitir casamento de padrões. Restam ainda duas propriedades, CONDICAO e ACAO. Na propriedade CONDICAO é armazenado um codigo especializado para detectar quando o LHS da regra é satisfeito. A confecção de um código especializado permite um processamento mais eficiente que se utilizassemos um código de aplicação geral para detectar casamento de padrões para qualquer regra. Na propriedade ACAO é armazenado o código especializado que irá criar uma cláusula quando executado, que será adicionada à memória de trabalho quando o LHS é satisfeito. Os códigos especializados assim armazenados ficam aptos para sua recuperação e execução a qualquer momento que se faça necessário utilizando o comando de LISP, "apply". Uma representação de uma regra de produção qualquer após o processo de codificação é mostrada na fig. V.1. Para implementar a codificação de uma regra segundo a estrutura mostrada criamos funções específicas para a execução das seguintes etapas :

a) Elaboração de código especializado para detecção de



predicado 1
PP1
CONDICAO
funcao lambda para deteccao de casamento de padroes
PX
PP2
predicado 2
ACAO
funcao lambda para ativacao do RHS
PPm
predicado m

FIG. V.1

"matching".

b) Atribuição/Atualização de valores às propriedades da regra e às propriedades dos atributos desta regra conforme foi mencionado anteriormente.

c) Elaboração de código especializado para adição de cláusulas à base de conhecimento (CONTEXTO) no caso de detecção de "matching".

A função CODIFICA-UMA-REGRA é a que faz a chamada das funções correspondentes a estas etapas. Para processamento da primeira e segunda etapa utilizamos a função auxiliar CODIFICA-CONDICAO que extrai da regra a lista contendo o LHS. Para processamento da primeira etapa, CODIFICA-CONDICAO invoca a função auxiliar CONSTROI-CONDICAO que apartir da constatação de quais os elementos (atributos) que se repetem em cláusulas diferentes do LHS da regra, constroi a função lambda aplicável a esta regra para detecção de "matching". Esta função por sua vez utiliza as funções auxiliares CONSTROI-TESTE e CODIFICA-PAR. CONSTROI-TESTE faz todas as combinações 2 a 2 necessárias das listas obtidas da extração dos atributos do predicado de cada uma das cláusulas do LHS. Cada par de listas obtido serve de parâmetro de entrada para a função CODIFICA-PAR que elaborará o segmento de codigo especializado para detecção de "matching" neste par de cláusulas. Cada segmento de código assim obtido é concatenado aos obtidos anteriormente para esta regra. No final do processo obtemos o código especializado completo para detecção de "matching" do LHS desta regra. Assim por exemplo para as cláusulas (AREA =AX), (QUENTE =AX) e (SENSIVEL-AO-CALOR =RX) que deram

origem as listas (=AX) (=AX) (=RX) com posições relativas numa regra hipotética iguais a 1, 2 e 3 é obtido o seguinte código :

```
(LAMBDA (NFACTS)
(EQ (NTH 0 (NTH 0 NFACTS))(NTH 0 (NTH 1 NFACTS))) )
```

que significa :

"O primeiro elemento da primeira lista de NFACTS é igual ao primeiro elemento da segunda lista de NFACTS ?"

Este código é atribuido como valor da propriedade CONDICAO da regra P5. NFACTS representa a lista de listas contendo os elementos aos quais será aplicada a função lambda de detecção de "matching" desta maneira criada. Seja esta lista p. ex. ((A2) (A2) (R1)) obtida das cláusulas (AREA A2), (QUENTE A2) e (ROTA R1); a aplicação da função lambda para esta lista irá devolver uma condição de "Verdadeiro" pois neste caso foi detectado casamento de padrões.

Para processamento da segunda etapa, CODIFICA-CONDICAO invoca a função auxiliar ATUALIZAR que fará a criação das propriedades da regra P5: PP1, PP2 e PP3; atribuindo a cada uma destas como valor os predicados das cláusulas teste correspondentes (1,2 e 3). Da mesma forma, para cada predicado de cada uma das cláusulas-teste no LHS da regra são criadas as propriedades PP1, PP2 e PP3 de acordo com a posição relativa da cláusula teste no LHS. Cada uma destas propriedades tem como conteúdo a lista de rótulos de regras onde foi detectada a cláusula teste naquela

posição. É necessária também a criação da propriedade LABEL-PROPS, associada a cada predicado do LHS contendo a lista dos nomes das propriedades associadas a este até o momento durante o processo de codificação de regras.

A terceira etapa (Elaboração de código especializado para adição de cláusulas à lista CONTEXTO que representa a parte da base de conhecimento representável por meio de cláusulas) é processada pela chamada da função CONSTROI-ACAO que é a função responsável pela geração do código especializado de adição de cláusulas. Esta função utiliza as funções auxiliares BUILD-ACTS e GERA-CODIGO. Seja o RHS do exemplo mostrado anteriormente para construção do código associado à propriedade CONDICAO, o seguinte :

(AREA-PROIBIDA =AX =RX 5)

O código gerado para ativação do RHS da regra P5 seria então o seguinte :

```
(LAMBDA (NFACTS)
(LIST 'AREA-PROIBIDA (NTH  0 (NTH  0 NFACTS))(NTH  0 (NTH  2
NFACTS))) )
```

Aplicar este código à lista NFACTS com valor igual a ((A2)(A2)(R1)) obtida dos fatos (AREA A2), (QUENTE A2) e (SENSIVEL-AO-CALOR R1), equivale a listar os átomos AREA-PROIBIDA, A2 e R1 para obter a representação simbólica da restrição (AREA-PROIBIDA A2 R1) que estabelece que A2 é uma área proibida para a rota R1. Este código gerado é associado à propriedade AÇAO da regra P5. O processamento das etapas descritas nesta seção permitirá um processamento mais eficiente do módulo de interpretação de regras descrito

na seção seguinte.

5.3 Interpretador de Regras.

Uma vez codificadas as regras podemos implementar o processo de interpretação de regras de uma maneira simples. O fluxo principal deste processo pode ser ilustrado pelo esquema da fig. V.2. O processo se inicia a partir do conjunto adição inicialmente igual ao conjunto CONTEXTO, contendo o conhecimento representável por meio de cláusulas do Sistema. O conjunto adição é o que contêm as cláusulas a serem adicionadas no ciclo 1 com a finalidade de obter instanciações de regras. É selecionada a cada ciclo, uma cláusula deste conjunto e são efetuadas todas as instanciações de regras possíveis. Durante este processo o conjunto conflito é incrementado com elementos contendo o rótulo de uma regra instanciada e a cláusula com possibilidades de ser adicionada ao conjunto CONTEXTO, como foi vista na seção anterior. Quando o conjunto adição é vazio, é ativado o ciclo 2 (Seleção de uma cláusula do conjunto conflito). Se esta seleção obtêm sucesso, o primeiro ciclo é reiniciado. Caso o conjunto adição e o conjunto conflito sejam vazios, o processo de interpretação de regras é finalizado.

A função INTERPRETADOR-DE-REGRAS é a que faz a chamada das funções de mais alto nivel deste processo. No ciclo 1 do Interpretador de regras, é invocada a função ENTRA-CLAUSULA que têm como objetivo obter todas as instanteações possiveis de regras após a seleção de uma regra do conjunto adição. Como resultado destas instanteações ENTRA-CLAUSULA obtêm os pares formados pelo

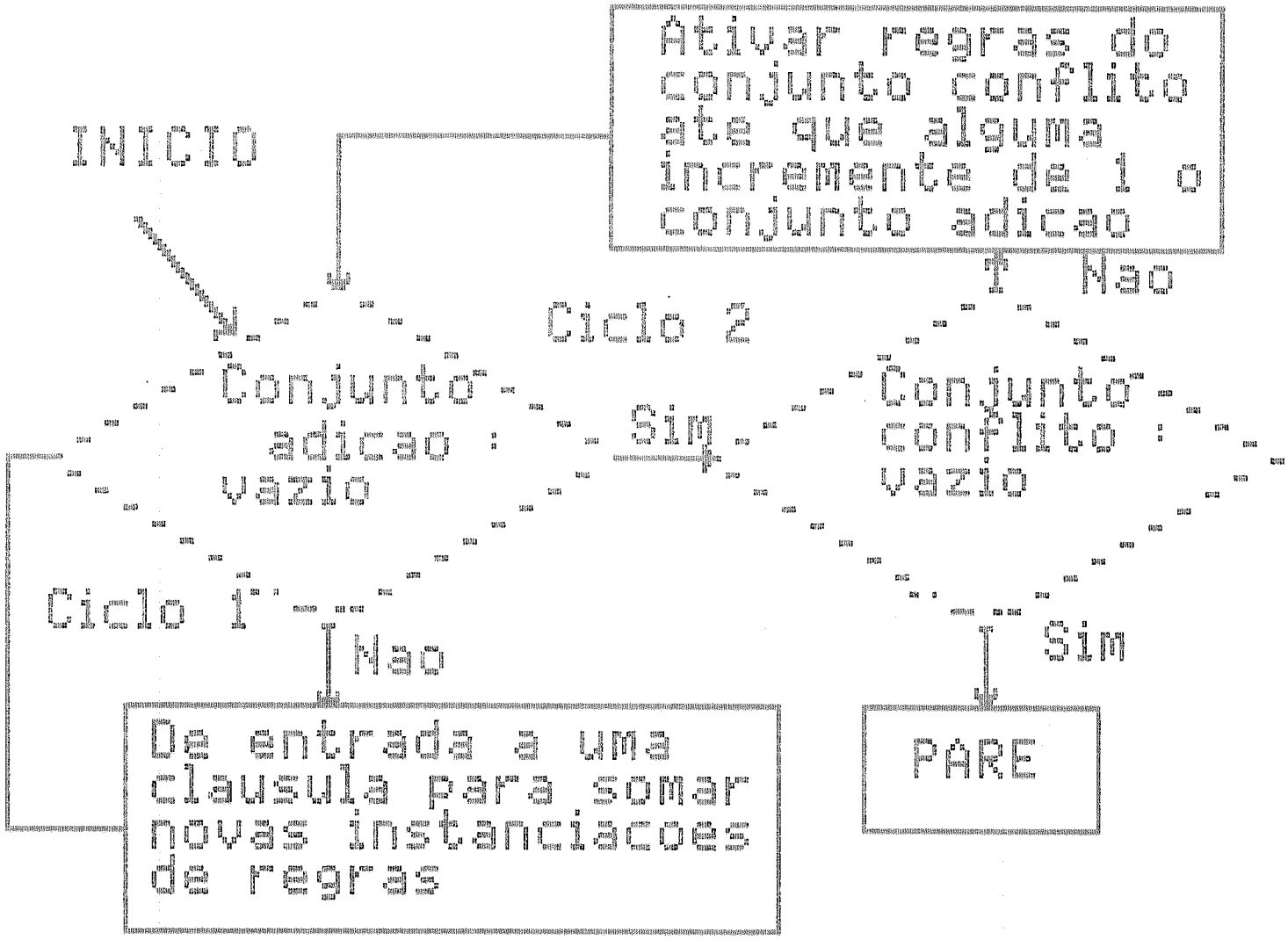

INICIO
Ativar regras do conjunto conflito até que alguma incremente de 1 o conjunto adição
Não
Ciclo 2
Conjunto adição vazia
Sim
Conjunto conflito vazio
Ciclo 1
Não
Sim
Dar entrada a uma palavra para somar novas instanciações de regras
PARE

FIG. V.2

rótulo da regra instanciada e a cláusula gerada pela ativação do RHS desta regra, que serão adicionados ao conjunto conflito atualizando o seu conteúdo. Assim por exemplo : (ENTRA-CLAUSULA (SENSIVEL-AO-CALOR R1)), é a chamada da função ENTRA-CLAUSULA com argumento (SENSIVEL-AO-CALOR R1) que é a cláusula que significa "A rota R1 é sensivel ao calor". Esta função retorna por exemplo os seguintes elementos :

(P1 (DISTANCIA-MINIMA R1 O2 5)), (P1 (DISTANCIA-MINIMA R1 O5 5))

que significa que a regra P1 foi instanciada duas vezes, gerando as declarações "A distância mínima da rota R1 para o obstáculo O2 é igual a 5" e "A distância mínima de R1 para o obstáculo O5 é igual a 5". A função ENTRA-CLAUSULA realiza também a atualização da propriedade denominada BINDINGS do predicado da cláusula entrada, pela adição ao conteúdo desta propriedade dos atributos do predicado desta cláusula. Desta forma, sempre que quisermos saber quais são os atributos de todas as cláusulas detectadas até o momento como possuindo o predicado da cláusula entrada, basta recuperar o conteúdo da propriedade BINDINGS deste predicado. Assim, a instrução em LISP:

(GET 'SENSIVEL-AO-CALOR 'BINDINGS) "Obter o conteúdo da propriedade BINDINGS do predicado SENSIVEL-AO-CALOR".

que devolve por exemplo a lista ((R1) (R3)), que significa que foi detectada até o momento a existência das cláusulas (SENSIVEL-AO-CALOR R1) e (SENSIVEL-AO-CALOR R3). A tarefa de

obter todas as instanteações de regras possíveis após a entrada de uma cláusula em ENTRA-CLAUSULA fica a cargo da função auxiliar MAPEA-CLAUSULAS. Esta última função é uma função recursiva que recebe inicialmente (primeiro nível de recursividade) como argumentos, os atributos do predicado, o predicado e o conteúdo da propriedade LABEL-PROPS (ver seção anterior) deste predicado. Cada elemento do conteúdo da propriedade LABEL-PROPS é indicativo das posições relativas onde o predicado apareceu nos LHS das regras codificadas no módulo de codificação de regras. Se este conteúdo é formado pelos elementos PP1, PP3 e PP4, significa por exemplo que em nenhuma regra do Sistema de Produçao, o predicado aparecerá na segunda posição do LHS, pois este foi detectado apenas nas posições 1 ou 3 ou 4 dos LHS de alguma(s) regra(s) do sistema.

A função MAPEA-CLAUSULAS utiliza a função auxiliar ENTER-NTHPROP que é a que de fato realiza a detecção de "matching". Para cada nome de propriedade obtido da recuperação do conteúdo de LABEL-PROPS, são feitas tantas chamadas da função ENTER-NTHPROP quantos rótulos de regras existirem no conteúdo desta propriedade do predicado. Assim, se por exemplo a recuperação da propriedade PP1 de um predicado devolve os valores P1, P4 e P5, isto quer dizer que o predicado acontece na primeira posição somente das regras P1, P4 e P5. Serão feitas neste caso, 3 chamadas da função ENTER-NTHPROP , uma para cada rótulo de regra recuperado. Desta forma a função ENTER-NTHPROP recebe como argumentos de entrada os atributos do predicado da cláusula entrada, um rótulo de regra onde o predicado existe e um nome de propriedade que indica a posição relativa no LHS da

regra da cláusula teste que contêm o predicado. Procedendo desta forma, são examinadas apenas as regras com alguma possibilidade de serem instanciadas.

Para entender melhor o processamento da função ENTER-NTHPROP, que desempenha um papel chave no processo de interpretação de regras acompanharemos o seguinte exemplo : Seja feita por exemplo a chamada de ENTER-NTHPROP com os argumentos R1, P5 e PP3. O argumento R1 provêm da cláusula (SENSIVEL-AO-CALOR R2) entrada em ENTRA-CLAUSULA, o argumento P5 indica que o predicado desta cláusula acontece na regra P5 e o argumento PP3 indica que a posição na regra P5, da cláusula teste que contêm o predicado da cláusula entrada é a terceira. Seja a regra P5 por exemplo a seguinte :

```
(P5 ((AREA =AX) (QUENTE =AX) (SENSIVEL-AO-CALOR =RX)) ==>
    (AREA-PROIBIDA =AX =RX))
```

que significa: "Se uma determinada área é quente e a rota planejada é sensivel ao calor, faça esta área uma área proibida para a rota".

O objetivo da função ENTER-NTHPROP é aplicar o codigo de detecção de "matching" contido na propriedade CONDIÇAO da regra P5 em todas as combinações de 3 elementos (a regra P5 possui somente 3 cláusulas teste) entre os elementos obtidos pela recuperação dos argumentos de todas as cláusulas existentes com o predicado AREA, os elementos obtidos pela recuperação dos argumentos de todas as cláusulas com o predicado QUENTE e a lista contendo os argumentos (R1) da cláusula entrada. Cada combinação deve possuir um elemento de cada um destes conjuntos. Sejam por

exemplo os conjuntos obtidos os seguintes : ((A1) (O4)), ((O4) (A1)) e ((R1)), as combinações possíveis são as seguintes :

((A1) (O4) (R1)), ((A1) (A1) (R1)), ((O4) (O4) (R1)) ((O4) (A1) (R1))

Cada uma destas combinações representam os valores que a lista NFACTS vista na seção anterior, irá assumir quando da aplicação do código de detecção de "matching". Ou seja, para cada uma das combinações obtidas, é feito um processamento do código especializado contido na propriedade CONDIÇAO de P5. Caso seja detectado casamento de padrões é ativado o código especializado de ativação do RHS, contido na propriedade AÇAO da regra. Finalmente são gerados os pares rótulo da regra combinada e cláusula gerada, a serem adicionados ao conjunto conflito. Um par gerado seria então por exemplo :

(AREA-PROIBIDA A2 R1) "A área A2 é uma área proibida para a rota R1".

O conjunto conflito é assim atualizado pela adição dos pares obtidos em cada processamento da função ENTRA-CLAUSULA. A função ENTER-NTHPROP utiliza nas diversas etapas que constituem o processo de detecção de "matching" que foi ilustrado acima, as funções auxiliares MULTI-CLAUSULAS, CRUZA-PRODUTO, CRUZA-AUXILIAR, MATCH-AND-EXECUTE e CHECK-FOR-DUPS. Cabe destacar a função auxiliar CHECK-FOR-DUPS que verifica se uma cláusula gerada pela ativação do RHS de uma regra já faz parte do conjunto CONTEXTO que contém a relação de fatos do Sistema de Produção. Desta forma é evitada a geração de cláusulas

repetidas.

A seleção de uma cláusula do conjunto conflito no ciclo 2 é feita mediante a aplicação de um critério de resolução de conflitos. Alguns módulos de inferência incorporam estratégias complexas de seleção. Em nossa implementação de módulo de inferência, é utilizado o critério simples de selecionar o primeiro elemento da lista que representa o conjunto conflito. Como foi mencionado no início desta seção, quando o conjunto adição e o conjunto conflito ficam vazios, é finalizado o processo de interpretação de regras.

5.4 Considerações finais sobre o Módulo de Inferência.

As vantagens das técnicas aqui usadas para implementação do módulo de inferência foram descritas inicialmente. O método descrito é suficiente para atender as necessidades do método de Planejamento de Rotas apresentado neste trabalho. A verificação da existência de uma restrição pelo módulo de avaliação de restrições contido no módulo de busca do caminho é feita simplesmente pela recuperação da propriedade BINDINGS da restrição que desejamos verificar. Se o valor retornado for nulo, não existe esta restrição para a rota. Caso contrário são retornados todos os argumentos de cláusulas que possuem este nome de restrição como predicado. Como foi mencionado inicialmente, o método aqui apresentado possui algumas características do algoritmo RETE apresentado em FORGY (1982). Um análise da complexidade para este algoritmo usado para implementação de um módulo de inferência para a linguagem OPS5 é mostrado naquele trabalho.

Capitulo VI - Revisão do Método de Planejamento de Rotas.

Ao fazer um anàlise do Método de Planejamento de Rotas no capitulo III deixamos varias questões em aberto. A validade das estruturas de Representação do conhecimento usadas por nosso Sistema foi abordada na seção 4.2.4. Neste capitulo será feita a revisão e/ou detalhamento de aspectos relacionados com a geração do espaço livre, tipo de restrições consideradas em nossa implementação do sistema, aperfeiçoamento do processo de utilização do conhecimento, funções de avaliação e heuristicas e possiveis extensões do Método de Planejamento de Rotas. No final deste capitulo serão apresentados detalhes de implementação do Sistema de Planejamento de Rotas.

6.1 Geração do espaço livre.

A representação da configuração baseada em celas obstáculo-adaptativas é uma representação melhor que a tradicional baseada em celas quadradas homogêneas que ocassionam um consumo elevado de memória e ineficiência do processo de busca do caminho. As celas obstáculo-adaptativas possuem ainda a vantàgem de conter informações sobre "vizinhos" da cela e "visões" desta que serão de grande utilidade na verificação feita pelo módulo de avaliação de restrições contido no módulo de busca do caminho. A descrição do algoritmo de geração de celas foi feita no capitulo III. Nesta implementação foi mantida este tipo de representação baseada em celas obstáculo-adaptativas permitindo porém, variações da representação espacial com a finalidade de aumentar a sua precisão. Nesta implementação

do módulo de geração de celas, é permitido um grau variável de precisão da representação gerada. Assim, previamente à geração de celas é passado pelo projetista a informação que indica o grau de precisão desejado.

GRAU 0 -> Aplicação normal do método de geração de celas.

GRAU 1 -> Aplicação do método de geração de celas sem a etapa de combinação.

GRAU 2 -> Aplicação da etapa de Geração modificada, para gerar como máximo 2^2*m celas, sendo m o número de celas obtido após a etapa de Geração pelo método original .

.
.
.

GRAU n -> Aplicação da etapa de Geração modificada, para gerar como máximo n^2*m celas, sendo m o número de celas obtido após a etapa de Geração pelo método original .

O aumento da precisão da representação pode ter utilidade quando a configuração problema é pouco densa em obstáculos e/ou quando o número de rotas a ser planejado é grande. Nestes casos, o número de celas gerado pode ser insuficiente para que o planejamento de rotas acabe em sucesso para todos os casos, apesar de que em algumas situações existe ainda espaço disponível nas celas que contêm segmentos de rota para conter outros segmentos pertencentes a outras rotas (Ver fig. VI.1).

A modificação da etapa de Geração a que nos referimos consiste em aumentar o número de pontos nos eixos coordenados que servirão de base para gerar inicialmente as

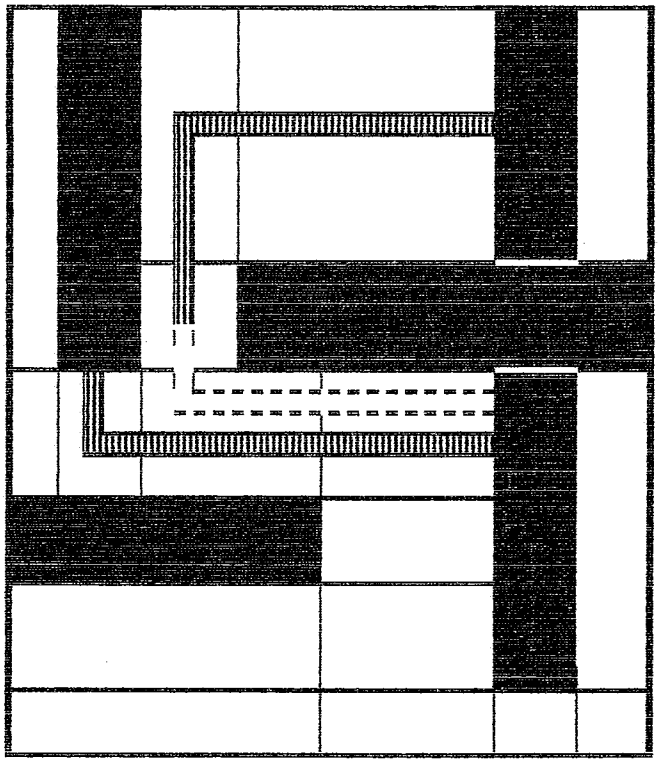


FIGURA VI.1

celas, previamente à etapa de Teste. Assim por exemplo, o grau de precisão 3 gera entre cada par de pontos de interseção com os eixos coordenados gerado pela prolonagação das arestas dos obstáculos, 3-1 = 2 pontos intermediários com espaçamento uniforme em relação aos pontos vizinhos.

Para um grau de precisão maior ou igual a 1, a etapa de Combinação é obviamente omitida, porque sua aplicação nos conduziria de volta à representação que seria gerada pela aplicação normal do método de geração de celas.

6.2. Descrição dos Principais tipos de restrições aplicáveis na busca do caminho e outras heuristicas sobre o problema.

a) Distancia Minima.

Esta restrição diz respeito ao espaçamento mínimo que deve possuir a rota de um obstáculo determinado quando ela passa na proximidade deste obstáculo. O valor desta distância é obtido pelo calculo da distância euclideana do centroide da cela pela qual passa a rota ao eixo imaginário do obstáculo paralelo à direção da rota na vizinhança deste substraido do cumprimento do raio r da rota.

A informação referente à este tipo de restrição pode ser incluida na base de conhecimento pelo projetista explicitamente ou através de inferência usando regras do seguinte tipo :

Regra em LISP :

((SENSIVEL-AO-CALOR =RX)(QUENTE =OX)(OBSTACULO =OX)) ==>

(DISTANCIA-MINIMA =RX =OX 5)

"Se a rota RX é sensivel ao calor e o obstáculo OX é quente, então o espaçamento minimo entre RX e OX deve ser de 5 unidades"

Esta restrição é do tipo rigido, ou seja, em nenhuma hipótese poderá ser violada. A possibilidade de quebra de uma restrição deste tipo é viabilizada através do uso de uma denominação diferente para a restrição como será visto a seguir. Esta restrição pode ser utilizado também em caso de incerteza sobre as dimensões e forma dos obstáculos da configuração. Assim, a declaração (DISTANCIA-MINIMA R1 O3 2) pode servir para a rota R1 manter um espaçamento minimo de 2 metros em relação ao obstáculo O3 se esta distância for considerada uma margem de segurança confiável dado o fato de haver incerteza nos dados referentes às coordenadas do obstáculo. A distância mínima "default" pode servir para a mesma de finalidade caso se deseje manter uma margem de segurança em relação a todos os obstáculos da configuração.

b) Distância Recomendada.

Esta restrição é definida de maneira idêntica à de distancia minima, mas neste caso não representa uma restrição rigida, tendo mais caracteristica de uma recomendação. Quando não satisfeita esta restrição o módulo de avaliação de restrições manipula esta informação de acordo com regras definidas pelo projetista para o relaxamento de restrições. Este conhecimento pode ser obtido através de regras como em a) ou fornecido explicitamente pelo projetista :

(DISTANCIA-RECOMENDADA R1 O3 4)

"A distancia recomendada da rota R1 para o obstáculo O3 é igual a 4 unidades"

c) Direção Proibida.

A direção que é proibida para a rota pode ser extraida de afirmações do seguinte tipo :

"Se a rota é sensivel a gravidade então a direção para cima é proibida para a rota."

ou fornecida explicitamente pelo projetista ao selecionar esta restrição dentre as opções oferecidas pelo programa que implementa o método de planejamento de rotas.

d) Vizinhança proibida.

Esta restrição diz respeito a um grupo de celas que representam uma área vizinha proibida e que não podem ser explorados pelo módulo de busca do caminho como segmentos de rota viáveis. A informação de se uma cela encontra-se dentro de uma vizinhança proibida é obtida por uma função de verificação especifica para esta restrição apartir de informaçao proporcionada pelo projetista. Podemos criar então a lista VIZINHANÇAS-PROIBIDAS com p.ex. o valor (O1 O3 ...). A função de verificação de restrições avaliará se a cela objeto de verificação tem como vizinhos um destes obstáculos. Caso afirmativo a cela encontra-se em área proibida. Da mesma forma que procedemos com a restrição "distancia-recomendada" podemos criar aqui a restrição "Area-Vizinha-Nao-Recomendada" de natureza não rígida.

e) Area Proibida.

Esta restrição estabelece áreas proibidas para passagem da rota. O projetista especifica a área proibida marcando no gráfico da tela de um video CRT, as celas que compõem a área na configuração de celas obstáculo-adaptativas gerada na primeira parte do método de planejamento de rotas. Conjuntamente o projetista informa o atributo destas áreas (p. ex.: quente, fria, umida) podendo também fornecer explicitamente a restrição. Quando esta última informação não está disponivel para o projetista, a informação referente ao atributo da área será tratada pelo módulo de inferência com base nas regras e fatos contidas na base de conhecimento como no seguinte exemplo:

Fatos em LISP :

(SENSIVEL-AO-CALOR R5) "A Rota R5 é sensivel ao calor"

(QUENTE A2) "A2 é quente"

(AREA A2) "A2 é uma área"

Regra em LISP :

((SENSIVEL-AO-CALOR =RX)(QUENTE =AX)(AREA =AX)) ==>
(AREA-PROIBIDA =RX =AX)

"Se a rota RX for sensivel ao calor e AX é uma área quente, então AX é uma área proibida para a rota RX"

Se houver alguma mudança na representação da configuração espacial, como é o caso na redefinição de celas abordado préviamente, deve ser feita uma redefinição destas áreas com base na nova representação.

f) Area Não Recomendada.

O procedimento descrito para AREA-PROIBIDA é

também valido aqui, com a ressalva de esta não ser uma restrição inviolável. Esta informação é extraida por exemplo de fatos e regras do seguinte tipo :

fatos :

(AREA A2)	"A2 é uma área"
(UMIDA A2)	"A2 é umida"
(SENSIVEL-A-UMIDADE R2)	"A rota R2 é sensivel á umidade"

Regra :

((UMIDA =AX)(SENSIVEL-A-UMIDADE =RX)(AREA =AX)) ==> (AREA-NAO-RECOMENDADA =RX =AX)

"Se AX é uma área, AX é umida, e a rota RX é sensivel ao calor, então AX é uma área não recomendada para a rota RX"

g) Áreas de Preferência.

O procedimento descrito em e) e f) para fornecer informações relativas à áreas serve também com ligeiras modificações para fornecer outro tipo de informação não restritivo, do tipo heurístico que ajudará a seleção de um caminho que possua o máximo possível as qualidades que se espera que a rota obtida possua. Pelo mesmo procedimento iterativo do projetista com a representação espacial gerada, podem então também ser definidas áreas que possuem certas características desejadas para a rota a ser planejada. Atributos possíveis da área definida são mostrados tais como refrigerada, ventilada, iluminada para selecionar os que são aplicáveis à rota.

A informação relativa à áreas de preferência pode por exemplo ser utilizada pelo módulo de busca do

caminho ao selecionar uma cela para expansão. Em caso de uma ou mais celas terem o mesmo "custo", que é o valor devolvido pela função de avaliação utilizada, a cela que pertencer a uma área de preferência, será preferida em relação a uma cela que não pertence à nenhuma área de preferência. Ou seja este tipo de informação é usada como criterio de desempate para seleção da cela a ser expandida. Em caso de persistir o empate de celas, o critério de desempate é arbitrário. Esta informação pode também ser usada como índice de performance, se a rota a ser planejada deve possuir o maior número de celas pertencente à áreas de preferência.

h) Segmento Mínimo.

Esta restrição diz respeito ao cumprimento mínimo que um segmento de rota deve possuir. No caso de projeto de tubulações por exemplo, esta restrição permite evitar que existam segmentos com cumprimento menor que o tamanho de uma tubulação padrão utilizada no projeto ou pode-se desejar também minimizar o número de "joelhos" utilizados. Um exemplo da declaração desta restrição é a seguinte :

(SEGMENTO-MINIMO R2 20) "O segmento mínimo da rota R2 é de 20 metros"

Análogamente, poderiamos definir a restrição "segmento mínimo recomendado" de natureza não rígida. Para cada cela avaliada por esta restrição são associadas a esta as informações sobre as coordenadas do seu centro de gravidade, o cumprimento atual do último segmento

determinado e a direção atual da rota. Caso seja detectada uma mudança de direção da rota é verificado se o último segmento é maior ou igual ao segmento mínimo requerido para a rota. Caso afirmativo é retornada uma condição de sucesso, caso contrário é retornada a condição de fracasso.

i) Cruzamento Proibido.

Esta restrição proibe explicitamente qualquer cruzamento entre rotas da configuração. Para verificar o cumprimento da restrição verifica-se se a cela candidata é uma das celas das listas de celas que compõem as descrições das rotas determinadas pelo sistema. Caso afirmativo, a cela não cumpre a restrição de cruzamento proibido. Análogamente, poderiamos definir a restrição "cruzamento não recomendado".

6.3 Função de avaliação e heurísticas.

Uma das afirmativas feitas em YASUHIRO (1986) é a de que o método de planejamento de rotas possui a capacidade de aceitar um número variado de restrições e índices de performance. Os aspectos relacionados com as restrições foram abordados no capítulo III e as principais restrições aceitas pela nossa implementação do método foram descritas na seção anterior. Não obstante não são dados detalhes de como seria viabilizada a manipulação de um número grande de índices de performance. O principal objetivo desta seção é mostrar a técnica utilizada em nossa implementação do sistema de planejamento de rotas para viabilizar o uso de uma função global de avaliação, seja esta simples ou composta.

A função de avaliação utilizada em

YASUHIRO (1986) pode ser considerada uma versão especializada do algoritmo A*. No capitulo IV foi visto que se h(n) <= h*(n), para qualquer nô n pertencente ao caminho determinado partindo do nô inicial s até o nô t pertencente o conjunto solução, então o A* é admissivel, isto é, é garantido achar o caminho ôtimo. Uma propiedade bem conhecida do A* estabelece que se existem duas funções heurísticas h_1 e h_2, tal que $h_1(n) < h_2(n)$ para todos os nôs de um grafo G, então o uso da heurística menos informada h_1 fará com que o A* expanda no mínimo, todo nô que seria expandido usando h_2. Porêm, o uso de uma versão do A* não admissível pode acarretar em alguns casos, em um aumento da eficiência do algoritmo, ocasionando entretanto a perda da garantia da otimalidade da solução. No capitulo IV vimos que muitas vezes no mundo real procuramos apenas uma boa solução sem estarmos preocupados com que esta seja ôtima. Em PEARL (1984) é mostrado que mesmo para o caso do uso de heurísticas não admissíveis, é possível garantir boas soluções sob certas condições. Outro resultado interessante apresentado aponta que a combinação aditiva g + h para h <= h*, é a maneira ôtima de agregar g e h para medida de custos aditivos. Assim, para estes casos, não faz muito sentido tentar outras combinações de g e h tal como p. ex.: f = g + h^2 /(g + h). Indices de performance, que servem para avaliar o "mérito" do caminho determinado no problema de planejamento de rotas abordado neste trabalho, tais como a métrica do caminho, tempo utilizado para percorrê-lo, custo em dinheiro necessário para construção do caminho, quantidade de pontos de preferência incorporados no caminho, etc! têm a caracterîstica de serem aditivos.

Conseqüentemente, sempre que o projetista deseje a solução ótima e ele disponha de uma heurística admissível deve ser utilizada a combinação aditiva g + h.

Como critério de implementação em nosso Sistema de planejamento de rotas, são aceitas pelo sistema, apenas combinações lineares de g e h do tipo:

$$f = v*g + w*h, \quad v,w \text{ pertencentes a } R^+.$$

A combinação em que v = w = 1, como foi dito anteriormente, é a ótima. Um sub-conjunto de combinações foi estudada por POHL (1970), quando v = 1 - w e w pertence ao intervalo [0,1]. Ele mostrou que se h <= h* então a combinação de g e h quando 0 <= w <= 1/2 conduz a uma busca admissível enquanto um valor de w > 1/2 pode conduzir a soluções sub-ótimas. Para os casos em que nos satisfazemos com a determinação de uma solução, seja esta ótima ou não, e nossa preocupação é com a minimização do custa da busca, o uso uso de w > 1/2 dá ao A* uma forte característica de busca em profundidade. Conclusões sobre experimentos do efeito de ponderar g e h são mostrados em PEARL (1984). Se for considerado vantajoso pelo projetista (considerações de Otimalidade X Eficiência), outras ponderações de g e h, para v e w pertencentes aos reais positivos podem ser tentadas e são aceitas pelo Sistema.

O uso de índices de performance e funções de estimação de custos variadas nos conduzem à utilização de funções compostas de g e h. A forma mais difundida na literatura de incorporar índices de performance compostos, é a combinação linear. A mesma estratégia será aqui usada para o caso de funções heurísticas compostas. Tomemos por

exemplo uma função global de avaliação F com m componentes :

(1) $F = F_1 + F_2 + \ldots + F_M$.

(2) $F = a_1 f_1 + a_2 f_2 + \ldots + a_m f_m$.

$F_1 \qquad F_2 \qquad\qquad F_M$

(3) $F = a_1(vg_1 + wh_1) + a_2(vg_2 + wh_2) + \ldots +$

$F_1 \qquad\qquad F_2$

$+ a_m(vg_m + wh_m)$.

F_M

(4) $F = (a_1 vg_1 + a_1 wh_1) + (a_2 vg_2 + a_2 wh_2)$

$F_1 \qquad\qquad F_2$

$+ \ldots + (a_m vg_m + a_m wh_m)$.

F_M

a_i =>coeficiente de f_i na combinação linear das diversas funções de avaliação que compõem a função global F.
a_i pertence a R^+, o conjunto dos números reais positivos.
i pertence a [1,m],
m pertence a N, o conjunto dos números naturais,

v,w =>coeficientes de ponderação de g e h.
v e w pertencem a R^+.

Seja por exemplo, a importância relativa das funções de avaliação que compõem a função global de avaliação a seguinte : $f_1 > f_2 > \ldots > f_n$; então verifica-se também que $a_1 > a_2 > \ldots > a_m$. Os coeficientes v e w são coeficientes de ponderação entre g e

h e são constantes para cada componente de g e h respectivamente. A forma padrão do A* ocorre para v = w = 1. A expressão (4) assume para este caso a seguinte forma:

$$(5) \quad F = \underbrace{a_1(g_1 + h_1)}_{F_1} + \underbrace{a_2(g_2 + h_2)}_{F_2} + \ldots + \underbrace{a_m(g_m + h_m)}_{F_M}.$$

Se cada coeficiente $a_i v$ e $a_i w$ da expressão (4) for representado por k_i e l_i respectivamente, é gerada seguinte expressão :

$$(6) \quad F = \underbrace{(k_1 g_1 + l_1 h_1)}_{F_1} + \underbrace{(k_2 g_2 + l_2 h_2)}_{F_2} + \ldots + \underbrace{(k_m g_m + l_m h_m)}_{F_M}.$$

O Sistema de planejamento de rotas indaga do projetista qual a ponderação de g e h, ou seja o valor dos coeficientes v e w. A seguir é pedida a informação de quais os pesos relativos de cada componente f_i da função global de avaliação. Finalmente, a partir destas informações o sistema gera a representação interna para a função global de avaliação na forma da expressão (6). A representação interna para cada elemento F_i, da função global de avaliação possui a seguinte estrutura :

$$((k_i, g_i), (l_i, h_i))$$

onde :

k_i => é o coeficiente a ser multiplicado ao valor

retornado pelo índice g_i. Este coeficiente é o resultado da multiplicação do coeficiente a_i do componente f_i e do coeficiente de ponderação v de g_i da soma ponderada de g e h.

g_i => Nome do índice de performance do componente f_i e que serve para fazer a chamada da função que avaliará a contribuição ao custo total do caminho determinado, realizada pela aplicação deste índice.

l_i => é o coeficiente a ser multiplicado ao valor retornado pela função heurística h_i. Este coeficiente é o resultado da multiplicação do coeficiente a_i do componente f_i e do coeficiente de ponderação w de h_i como consequência da soma ponderada de g e h, como foi visto anteriormente.

h_i => Nome da função heurística de estimativa de custo utilizada pelo componente f_i. Caso não disponhamos da informação necessária que permita a utilização de uma função heurística, h_i assume o valor do elemento nulo (NIL).

A representação interna da estrutura de uma função de avaliação na forma da expressão (6) é feita por meio de listas. A representação em LISP da função de avaliação exemplo desta seção é a seguinte :

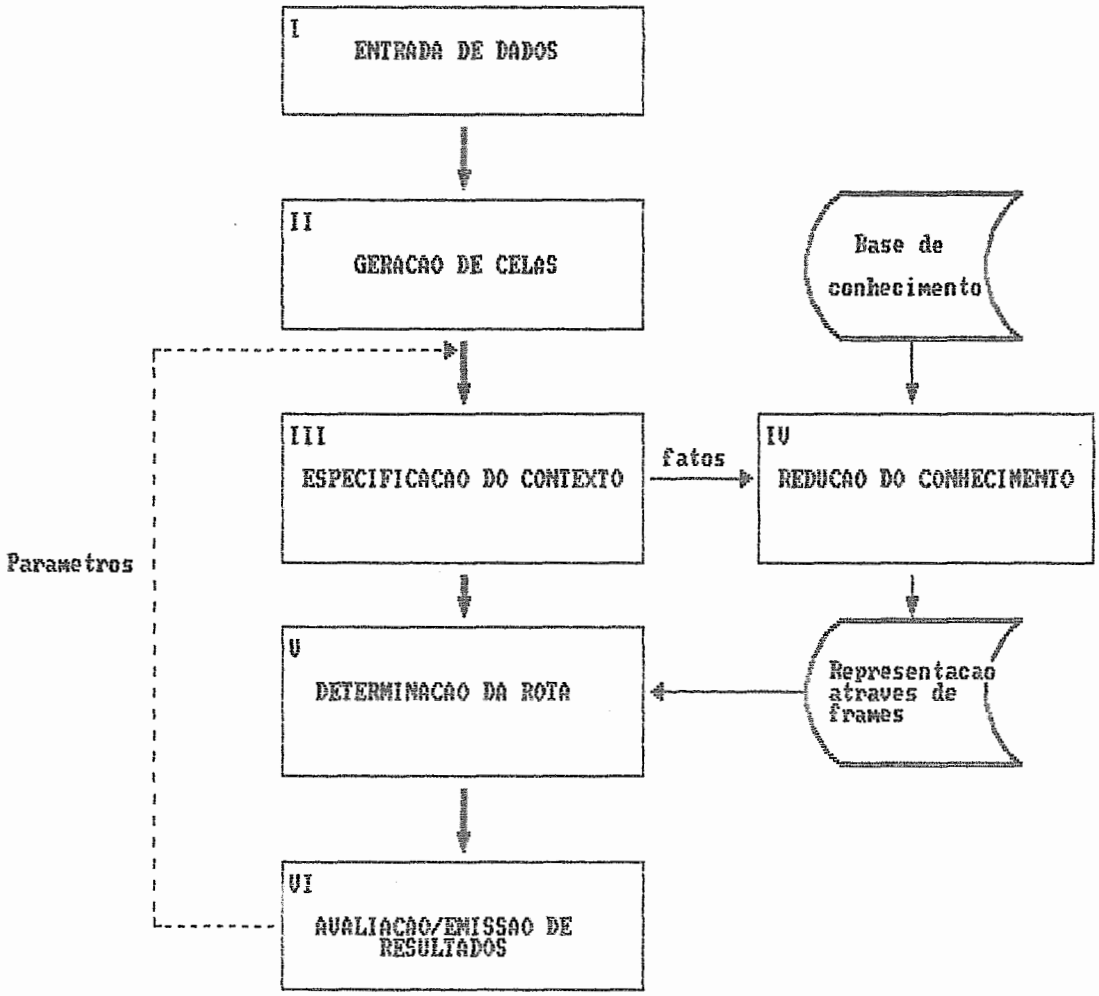
(((k1 g1)(l1 h1)) ((k2 g2)(l2 h2)) ... ((km gm)(lm hm))).

Cada elemento desta lista representa um elemento F_i da função global de avaliação F.

Na etapa III, de especificação do contexto da rota mostrada na figura VI.2, são dadas ao projetista, as opções de índices de performance e funções heurísticas disponíveis no Sistema para que este possa especificar os diversos componentes g_i e h_i e os seus coeficientes a_i. Como já foi visto, aqui o projetista especifica também os coeficientes de ponderação v e w de g e h. O Sistema gera então a representação interna da função global de avaliação. O módulo de Redução do conhecimento recebe esta representação interna e a transporta para o slot "indices-de-performance" do frame da rota que está sendo planejada. O algoritmo de busca do caminho utiliza a informação contida no slot "indices-de-performance" para invocar as funções em LISP necessárias para calcular o valor a ser retornado pela função global de avaliação quando uma cela é atingida durante a busca. A estrutura modular do Sistema permite que para a inclusão de novos índices de performance e funções heurísticas, seja necessário apenas a criação de funções específicas correspondentes a cada um destes novos índices de performance e heurísticas sem precisar alterar o controle global do Sistema e sem precisar revisar os outros módulos. Esta modularidade foi em parte conseguida pela implementação do Sistema em uma linguagem funcional como é o LISP.

6.4 Utilização do conhecimento pelo Algoritmo de Busca do Caminho.

O mecanismo de utilização do conhecimento já foi visto no Capitulo III. A verificação de se uma cela cumpre determinada restrição do problema é feita pelo módulo



MODULOS DO SISTEMA DE PLANEJAMENTO DE ROTAS

F I G U R A VI.2

de avaliação de restrições do algoritmo de busca do caminho. As informações obtidas na avaliação de cada restrição individualmente, tem validade apenas para a rota que está sendo determinada e assim mesmo apenas são armazenadas informações que servirão para avaliação em celas vizinhas de restrições em que o caminho anterior da rota é importante, tal como nos casos de "direção proibida" e "segmento mínimo". Sabemos que uma cela pode ser atingida varias vezes durante a busca e que dependendo da restrição a ser avaliada, pode ser oneroso computacionalmente o processo de verificação de viabilidade da cela. Seria conveniente neste caso incorporar definitivamente na busca todo o conhecimento obtido previamente durante o processo de resolução do problema. Se o planejamento de rotas envolve a determinação de mais de uma rota, então esta estratégia tem uma justificação ainda maior, pois os espaços de busca utilizados na determinação das várias rotas tem a possibilidade de se intersectar. O termo "espaço de busca" se refere ao conjunto de celas atingidas durante a busca da solução.

O uso da estratégia de incorporação do conhecimento obtido durante a busca está limitado, entretanto, pelas seguintes condições :

a) O conhecimento obtido por uma determinada função componente do módulo de avaliação de restrições, deve ter possibilidades de ser re-aproveitado posteriormente, isto é, se este conhecimento é válido apenas para condições muito particulares, que dificilmente se repetirão, a adoção desta estratégia não é recomendável

e pode até ocasionar um aumento do custo computacional.

b) A incorporação de conhecimento realizada por uma função de avaliação de restrição deve trazer algum benefício do ponto de vista da eficiência computacional. Isto quer dizer que não podemos obter vantagens deste processo em funções de avaliação de restrições que fazem verificações simples, pois correríamos o risco de aumentar a complexidade do processo de verificação que já é por natureza, simples. A vantagem do processo de incorporação do conhecimento é obtida quando do seu uso em funções de verificação complexas, onerosas computacionalmente.

Com relação à condição b), as restrições tratadas por nossa implementação do Sistema de Planejamento de Rotas e abordadas na seção 6.2 podem ser consideradas como do tipo simples e conseqüentemente não usaremos nenhum esquema de incorporação do conhecimento na implementação do Sistema.

Existem algumas restrições menos usuais no problema de planejamento de rotas e que são mais difíceis de serem avaliadas nas quais se justifica a estratégia de incorporação do conhecimento. Como exemplo, podemos citar a restrição que denominaremos DISTANCIA-EUCLIDEANA. Esta restrição diz respeito à distância mínima euclideana que uma rota deve manter em relação a determinados obstáculos da configuração. Depois do processamento do Módulo de Redução do conhecimento a informação referente à distancia euclideana aparece na base de conhecimento na forma de declarações do seguinte tipo :

"A distância euclideana a ser mantida da rota R1 para o obstáculo O3 é de 3 unidades no mínimo"

Declaração em LISP : (DISTANCIA-EUCLIDEANA R1 O3 3)

Um procedimento para verificação da viabilidade da cela pela função de verificação de distância euclideana seria o seguinte :

a) Classificar em ordem crescente da distância DRO os obstáculos da configuração. A distância DRO é a distância euclideana do obstáculo para a rota computada como a distância do obstáculo para o centroide da cela menos o raio da rota.

b) Para cada declaração de distância euclideana contida na base de conhecimento faça :

   b.1 Pesquisar sequencialmente a lista classificada pela distância DRO até que seja achada a distância do segmento de rota para o obstáculo a que a declaração se refere ou até que a distância associada ao elemento da lista de distancias seja maior que a distância mencionada na declaração de distância euclideana.

   b.2 Se a pesquisa finalizou pela primeira condição de parada de b.1, marcar a cela como não viável e finalizar o processamento da função de verificação.

A verificação da restrição DISTANCIA-EUCLIDEANA é onerosa computacionalmente principalmente por causa do passo a), e é um exemplo do caso em que a utilização do processo de incorporação do conhecimento é

vantajosa. A incorporação do conhecimento pode ser realizada de uma maneira muito simples em LISP. Através de listas expressamos simbólicamente a informação obtida por uma função especifica de verificação da restrição. No caso da distância euclideana a função de verificação é modificada para gerar no final do seu processamento uma das seguintes declarações de acordo com a condição de parada :

(VERIFICACAO-DE CX 'SUCESSO)

"A verificação da restrição de distância euclideana para a cela CX obteve sucesso"

(VERIFICACAO-DE CX 'FRACASSO)

"A verificação da restrição de distância euclideana para a cela CX terminou em uma condição de fracasso"

Outros fatos que sirvam para reconhecer o contexto em que a cela foi avaliada por uma função de verificação também devem ser armazenados. Cada função de verificação que incorpore conhecimento deve acessar então listas de fatos associadas às celas já avaliadas da configuração, para verificação inicial antes de iniciar o processamento da função propiamente dito.

Uma ültima modificação é necessária na especificação do contexto da rota (Passo 3 na figura VI.2). Se a rota a ser planejada possui propriedades diferentes das propriedades da rota anteriormente determinada, o conhecimento anteriormente obtido na busca não é mais válido e conseqüentemente as listas associadas às funções de verificação que incorporam conhecimento serão "limpadas". Um

conjunto de regras é incluido no módulo de especificação de contexto para verificar a identidade de propriedades entre rotas. As regras são do seguinte tipo :

"Se as informações referentes ao tipo da rota são diferentes do da rota anterior, atribuir à variável TIPO o valor 'não-identicas' ".

"Se a largura da rota a ser planejada é diferente da largura da rota anterior, atribuir à variável TIPO o valor 'não-identicas' ".

"Se o conteúdo da variável TIPO é 'não-identicas' limpar as listas associadas às funções de verificação que incorporam conhecimento".

6.5 Relaxamento de restrições.

O tipo de restrições utilizado em YASUHIRO (1986) estabelecem uma clara divisão entre as celas da configuração espacial : as viáveis e as não viáveis. As celas viáveis são aquelas que satisfazem as restrições do problema. Esta verificação é feita pelo módulo de avaliação de restrições do módulo de busca do caminho. As celas não viáveis são definidas pelo negativo, ou seja são aquelas que não satisfazem pelo menos uma das restrições do problema. Porém existirão situações em que quebrar uma determinada restrição ou restrições pode ser interessante, seja por causa da possibilidade de gerar uma rota muito mais econômica (de acordo com os critérios da função global de avaliação utilizada), seja pela razão de que a seleção da cela poderia ser considerada vantajosa de acordo com o

conhecimento heurístico que o projetista possui. Para viabilizar a quebra de restrições, o sistema de planejamento de rotas deve possuir um mecanismo para relaxamento de restrições.

O relaxamento de restrições se dá em duas etapas. A primeira etapa é a nivel de programa. O Módulo de Reduçäo do conhecimento obtêm o conhecimento primitivo através de inferências a partir da base de conhecimento. Säo obtidas nesta etapa duas classes de restrições, as de natureza rígida e as näo rigidas. Ao se fazer esta separação, ficam determinadas as restrições com possibilidades de serem relaxadas e as que näo poderäo ser relaxadas em hipótese nenhuma.

A segunda etapa envolve interação com o projetista. Para entender a forma em que se dá esta interação, descreveremos primeiro de forma genérica o algoritmo utilizado na implementação do módulo de avaliação de restrições.

MODULO DE AVALIAÇAO DE RESTRIÇOES.

==========

-Atribuir à cela o estado "viável".

-Para cada restrição do slot "restrições" do frame da rota faça :

Invocar a função de verificação correspondente.

Se for detectada quebra de restrição :

Se a restrição for do tipo rígido então faça :

Mudar o estado da cela para "näo viável".

Finalizar o processamento do módulo.

Senão faça :

Mudar o estado da cela para "temporàriamente não viàvel".
Armazenar os valores obtidos pela função de verificação e o nome da restrição infringida.

Fim-se.

Fim-se.

-Mostrar na tela o gràfico da representação baseada em celas da configuração, estàgio da busca, a cela atingida, posição inicial e posição da cela objetivo, caminho parcialmente determinado, restrições que foram infringidas assim como os valores obtidos pelas funções de verificação.

-Pedir do projetista a confirmação do relaxamento das restrições para esta cela.

-Se o projetista confirmou o relaxamento então faça :

Atribuir à cela o estado de "viàvel".
Finalizar o processamento do mòdulo de avaliação.

Senão faça :

Atribuir à cela o estado de "não viàvel"
Finalizar o processamento do mòdulo de avaliação.

Fim-se.

Desta forma o processo de busca do caminho torna-se conversacional e o sistema mostrarà gràficamente a representação espacial da configuração e uma sèrie de

informações úteis obtidas durante a busca para o projetista poder fazer a avaliação da conveniência da viabilização de uma determinada cela com possibilidades de conter um segmento de rota. Uma vantagem desta abordagem é a natureza altamente flexível do processo decisório. Um preço a ser pago é o aumento das iterações homem-máquina, embora com muita vantagem ainda em relação aos métodos tradicionais que não incorporam técnicas de representação do conhecimento.

6.6 Representação e utilização do conhecimento em outros problemas de planejamento de rotas.

A arquitetura modular do Sistema de planejamento de rotas ilustrado no capítulo III revela uma certa independência entre o método de geração da representação do espaço livre e os esquemas de redução do conhecimento, independência esta que se dá também em relação ao processo de geração de frames e à determinação da rota. Isto nos leva a pensar que seriam necessárias relativamente poucas alterações nas técnicas aqui apresentadas para adaptá-las à solução de outros tipos de planejamento de rotas entre obstáculos. As adaptações poderiam ser feitas encima de métodos de planejamento de rotas já existentes tais como por exemplo, os apresentados no capítulo II e III desta tese. Especificamente nos interessam os métodos que utilizam celas como forma de representação do espaço livre da configuração. Como foi dito no capítulo II na descrição do método de Lozano-Perez e constatado por nós na implementação do Sistema de planejamento de rotas, a utilização do complemento dos obstáculos, chamado de espaço livre facilita o uso de variadas heurísticas de busca.

Comentaremos a seguir possíveis adaptações de alguns métodos de busca de caminho entre obstáculos de modo a estes permitirem a utilização das técnicas de representação e utilização de conhecimento descritas nesta tese.

METODO DE LOZANO-PEREZ E BROOKS.

Estes métodos tem sua principal aplicação em robótica e permitem a movimentação de objetos sujeitos ou não a rotações. A principal restrição tratada por ambos métodos é quanto a evitar posições sujeitas a colisões o que é resolvido pelo computo do espaço livre da configuração. O uso das técnicas de representação e utilização do conhecimento descritas nesta tese tornam estes métodos aptos a receber um número maior de restrições, heurísticas e funções de avaliação assim como a entrada do conhecimento em um nível mais elevado. A arquitetura de um Sistema de planejamento de rotas que tome como base estes métodos é idêntica à descrita nos capitulos III e VI (Ver figura VI.2), resalvando pequenas modificações segundo o tipo de problema abordado :

Etapa de geração de celas : São geradas celas poliédricas no método Lozano-Perez e cônicas no método de Brooks. O processo de geração de obstáculos e celas poliédricas é descrito em LOZANO-PEREZ (1979, 1981 e 1983) e o de celas cônicas em BROOKS (1983).

Restrições e heurísticas sugeridas utilizadas : Podem ser do mesmo tipo que as neste capítulo ou acrescentar outras se necessário, porém terão que ser implementadas levando-se em consideração a forma e dimensões dos objetos da configuração.

Funções de avaliação : Na construção de funções de avaliação deve se levar em consideração a forma e dimensões dos objetos da configuração.

É recomendado o uso de frames com estrutura básica idêntica à utilizada por nosso sistema (Ver seção 6.8.1), podendo ser acrescentados slots segundo conveniência.

O módulo de inferência aqui desenvolvido é de utilidade geral e pode ser utilizado em quaisquer processos de inferência onde as clausulas e regras utilizem estruturas de representação idênticas às descritas no capitulo V.

O algoritmo de busca do caminho pode ser básicamente a versão do A* aqui utilizada ressalvando que as ligações entre celas vizinhas tem no método de Lozano-Perez como ponto intermediário, o centroide da interseção destas. Esta medida têm, como foi visto no capitulo II, a finalidade de evitar que arco entre duas celas vizinhas passe por fora destas ocorrendo sub-estimação do comprimento do arco que as une. São consideradas neste caso, celas vizinhas, todas aquelas que têm pelo menos um ponto ou vértice em comum. No método de Lozano-Perez, o cômputo

de distâncias entre o objeto movimentado e obstáculos da configuração a fim de evitar colisões, toma como base um ponto de referência rv_a no objeto, rv_a pode servir de base para avaliação de outras restrições que levem em consideração distâncias entre o objeto movimentado e um outro objeto qualquer da configuração. Como sabemos que em geral é mais complexa a computação envolvida no cálculo de distâncias entre objetos com dimensões que no caso de distância entre dois pontos, pode-se tornar viavel a assimilação do conhecimento obtido durante a determinação da rota tal como sugerido na seção 6.4.

A incerteza inerente aos problemas reais, e que em nosso caso se refere às formas e dimensões dos objetos da configuração, podem ser tratadas fazendo pequenas sobre-estimações nas dimensões destes. Algumas técnicas para lidar com este tipo de incerteza são sugeridas em LOZANO-PEREZ (1979) e existem outras na literatura especializada.

VARIAÇOES DO SISTEMA DE PLANEJAMENTO DE ROTAS.

O Sistema aqui implementado permite o planejamento de rotas Manhattam. Com pequenas alterações poderiamos adaptá-lo para a solução de problemas de determinação de rotas não Manhattam e também de modo a levar em consideração a forma e dimensões de um objeto a ser movimentado pela rota determinada. Assim no módulo de busca do caminho poderiamos modificar o procedimento de expansão de modo que ao expandir uma cela fossem atingidas todas as celas que tivessem pelo menos um vértice em comum com a cela expandida. Isto permite que no caso do problema

bi-dimensional possam ser atingidas até 8 celas vizinhas e no caso tri-dimensional até 26 celas. A ligação entre celas seria unindo-se os centro de gravidade destas, sendo porém necessário a criação de um ponto intermediário na ligação entre celas coincidindo com o centroide da interseção destas.

No caso de ser necessário levar em consideração a movimentação de um objeto pela rota podemos aproximar a projeção do objeto em duas dimensões (no problema bi-dimensional) a um círculo com diâmetro igual à maior distância entre dois pontos do objeto. Esta abordagem nos permite com que possamos lidar também com rotações do objeto. Neste caso um cela será considerada viável se entre outras coisas, a largura da cela for maior ou igual ao diametro do envoltório circular determinado para o objeto. Para o problema tri-dimensional esta mesma abordagem nos leva à criação de envoltórios esféricos.

6.7 Sistemas Especialistas.

O Sistema de planejamento de Rotas aqui desenvolvido serviu para testar as técnicas descritas nesta tese e avaliar o aperfeiçoamento destas. A base de conhecimento original do Sistema engloba regras de produção aplicáveis aos problemas de projeto de tubulações em geral, canais, projeto de layout de placas VLSI e projeto de layout de plantas industriais. Embora fosse uma preocupação nossa a de dotar o sistema da capacidade de resolver os principais tipos de problemas que surgem no planejamento de rotas nestas áreas, o objetivo principal ao fazer a implementação não foi a de obter um sistema de I.A. que fosse considerado

como "pronto" para aplicação em qualquer uma destas áreas, mas testar as técnicas chaves necessárias à implementação de qualquer sistema de I.A. destinado a resolver problemas de planejamento de rotas nas áreas propostas. Assim, poderiamos obter um sistema especialista com base em nosso sistema de planejamento de rotas através de interação com peritos no assunto em áreas de aplicação específicas. Seria possível ainda o desenvolvimento de sistemas especialistas com bases de conhecimento mais abrangentes em que o problema de determinação de rotas fosse apenas um dos problemas a serem resolvidos e os dados obtidos na resolução de um problema ajudassem a construir a base de conhecimento a ser utilizada na resolução de outro. Para exemplificar tomemos como referência o problema do projeto de tubulações industrias. As informações sobre projeto de tubulações descritas a seguir foram obtidas em SILVA TELES (1987). As tubulações industriais tem a seguinte classificação segundo a classe de emprego :

Tubulações Industriais :

- Tubulações dentro de instalações industriais :
  - Tubulações de processo
  - Tubulações de utilidades
  - Tubulações de instrumentação
  - Tubulações de transmissão hidráulica
  - Tubulações de drenagem
- Tubulações fora de instalações industriais :
  - Tubulações de transporte (adição, transporte, drenagem)

- Tubulações de distribuição (distribuição, coleta, drenagem)

De esta classificação reduzamos nosso escopo ainda para o caso de Tubulações de processo. O projeto de tubulações de processo envolve entre outros os seguintes aspectos :

- Seleção de tubos, materiais, processos de fabricação e diâmetros e espessuras de tubos.
- Meios de ligação entre tubos.
- Seleção e alocação de valvulas.
- Seleção e alocação de acessórios de Tubulação.
- Alocação de Juntas de Expansão.
- Escolha e dimensionamento de purgadores de vapor, separadores diversos e filtros.
- Disposição das construções em uma Instalação Industrial.
- Traçado e Detalhamento de Tubulações.
- Desenhos de Tubulações.
- Critérios de projeto para suportes de Tubulação.
- Aquecimento, Isolamento térmico, Pintura e Proteção.
- Cálculo do Diâmetro de Tubulações.
- Esforços e Tensões que atuam sobre tubos.
- Cálculo de vão entre suportes.

Dentro dos aspectos mencionados poderiamos relacionar apenas como amostra da possibilidade da construção de Sistemas Especialistas nesta área, o problema da seleção de materiais. São os seguintes, entre outros, os fatores que influem na escolha de materiais :

- Condições de serviço (pressão e temperatura de trabalho).

- Fluido conduzido : Natureza e concentração do fluido, impurezas, pH, resistência à corrosão do material, possibilidade de contaminação do fluido pelos resíduos da corrosão, etc.
- Nível de tensões suportadas pelo material.
- Natureza dos esforços mecânicos : Tração, compressão, flexão, vibrações, etc.
- Custo do Material.
- Segurança.
- Experiência prévia.

É possível nos perguntar-mos : Seria suficiente utilizar para a realização destas tarefas relacionadas com o projeto, apenas dados constantes em manuais de normas técnicas e na bibliografia especializada?. Quem responde a esta pergunta é Silva Teles, engenheiro especializado em projeto de tubulações em SILVA TELES (1976):

"Para a solução do problema de seleção de materiais, a experiência do projetista (ou da organização de projetos) é indispensável e insubstituível. Só a experiência, resultado do acúmulo de soluções adotadas em casos anteriores, é capaz de julgar com objetividade e segurança o grau de influência de cada um dos fatores acima (os que nós acabamos de mencionar) ".

Assim por exemplo, na hipotese da construçao de um sistema especialista em seleção de materiais e planejamento de rotas poderiamos ter entre outras, regras do

seguinte tipo :

"Se o fluido transportado é um hidrocarboneto com baixo enxofre (até 1%) e o material do tubo é aço-carbono, então o tubo é sensivel a calor acima de 320°C e a espessura deve ser de 1,2 mm."

"Se o fluido transportado é um hidrocarboneto com baixo enxofre (até 1%) e o material do tubo é aço-liga 5Cr-1/2Mo, então o tubo é sensivel a calor acima de 400°C e a espessura deve ser de 1,2 mm."

"Se o fluido transportado é um hidrocarboneto com enxofre normal (1% a 3%) e o material do tubo é aço-carbono, então o tubo é sensivel a calor acima de 280°C e a espessura deve ser de 1,2 mm."

"Se o fluido transportado é um hidrocarboneto com enxofre normal (1% a 3%) e o material do tubo é aço-liga 5Cr-1/2Mo, então o tubo é sensivel a calor acima de 350°C e a espessura deve ser de 1,2mm."

"Se o fluido transportado é agua doce e o material do tubo é aço-carbono galvanizado e o tubo tem até 4" de diâmetro, então o tubo é sensivel a pressões acima de 10 kg/cm² e a temperaturas acima de 60°C".

Poderiamos integrar estas regras com outras relacionadas exclusivamente com o problema de planejamento de rotas como por exemplo :

"Se o problema é de projeto de tubulações e o tubo é sensivel a calor acima de 350°C, então a rota é sensivel

a calor acima de 350°C."

"Se a rota é sensivel a calor acima de 350°C e um obstáculo é quente com temperatura acima de 350°C, então a distância mínima da rota para o obstáculo deve ser de 5 metros".

O levantamento do traçado e detalhamento da tubulação por parte do projetista pode levar à criação de restrições de distância mínima a serem mantidas entre tubos. Elevações a serem mantidas pelas tubulações podem ocasionar à geração da restrição de que determinadas rotas se mantenham dentro uma determinada cota de elevação para serem consideradas viáveis. A tarefa de calcular o diâmetro do tubo pelo projetista nos levaria no caso da construção de um sistema especialista de planejamento de rotas para tubulações a regras do seguinte tipo :

"Se o problema é de projeto de tubulações, o diâmetro calculado para o tubo é de X e a espessura de Y, então a largura da rota é igual a X mais duas vezes Y."

Existem ainda outras regras que poderiam fazer parte da base de conhecimento do sistema dependendo dos aspectos do projeto que se pretende resolver com o sistema e/ou da finalidade principal do Sistema, seja esta o projeto de tubulações, canais , layout de plantas industriais ou circuitos VLSI ou outro problema afim.

6.8 Implementação do Sistema de Planejamento de Rotas.

O objetivo perseguido ao realizar a implementação do método de planejamento de rotas foi o de

testar o operacionalidade das ideias expostas neste trabalho, observar a eficiência do processo de planejamento de rotas como um todo assim como a qualidade das rotas obtidas. Um dos problemas com que nos deparamos ao construir Sistemas de Produção em áreas onde nem todo o conhecimento pode ser considerado como do tipo heurístico tais como na Engenharia, consiste em como lidar com conhecimento representado por procedimentos formais, rígidos e com conhecimento heurístico simultâneamente. Este também foi o caso na construção do Sistema de Planejamento de Rotas. No planejamento de rotas são utilizados conhecimentos sobre procedimentos para geração da representação espacial, realização de inferências a partir da base de conhecimento mantida pelo Sistema, geração de estruturas de representação do conhecimento (Frames) e algoritmo de determinação da rota. A geração da representação espacial por exemplo, envolve o conhecimento de um procedimento do tipo rígido porém invariável pois mesmo em contextos diferentes o procedimento de geração de celas será sempre o mesmo. O mesmo pode-se dizer do processo de geração de frames. Dados como coordenadas dos vértices dos obstáculos e do layout problema, assim como a descrição dos pontos inicial e final da rota e da largura desta são dados do problema do tipo numérico que estamos acostumados a tratar. Por outro lado, a descrição detalhada da rota e dos obstáculos feita na forma de clausulas de lógica de predicados, assim como de regras de produção para obtenção de novas clausulas representam o conhecimento do tipo heurístico sobre o problema.

O Módulo de Inferência desenvolvido para o Sistema permite que possamos recuperar da base de

conhecimento após a realização de todas as inferências possíveis previamente determinação da rota, informações tais como os tipos de restrições e heurísticas aceitos pelo Sistema e os que serão aplicados à rota. Este tipo de informação como já vimos no capitulo III, constitue o conhecimento de mais baixo nivel e é denominado por nós de conhecimento primitivo, que é o que efetivamente será utilizado pelo Módulo de busca do caminho na determinação da rota. O conhecimento sobre o contexto compreende então dados numéricos e clausulas de lógica dos predicados. A integração deste conhecimento de diferente natureza é feita usando frames como estrutura de representação do conhecimento intermediária a ser manipulada pelo Sistema. Nesta estrutura funções de avaliação de performance e verificação de restrições são tratadas como dados. Por exemplo, no slot "Restrições" do frame da rota aparecem somente o rótulo das restrições que são aplicáveis ao problema de planejamento da rota corrente e não todas as restrições aceitas pelo Sistema. O Módulo de Avaliação de Restrições do Módulo de busca do caminho fará somente a invocação das funções de verificação de restrições das restrições constantes no slot "Restrições". Para cada restrição que aparece no slot "Restrições" o sistema criará um slot correspondente no frame da rota contendo os conjuntos de termos pertencentes à clausulas cujo predicado é o nome da restrição e um dos seus termos é o rótulo da rota que está sendo determinada. Nesta seção apresentaremos o algoritmo de geração do slot "Restrições" como ilustração do processo de geração de frames pelo Sistema. O agrupamento do conhecimento a ser utilizado pela rota na determinação do caminho permite que

possamos dar um tratamento uniforme e eficiente aos diversos tipos de informações que dispomos, recuperando ou armazenando rápidamente dados nos slots do frame da rota. A descrição das estruturas de dados usadas pelo sistema é abordada na seção 6.6.1. A descrição da arquitetura geral do Sistema de Planejamento de Rotas foi realizada no Capitulo III, a descrição da arquitetura e implementação do Módulo de Inferência foi feita amplamente no Capitulo V, e a do algoritmo do sub-módulo de avaliação de restrições do módulo de busca do caminho, na seção 6.5 deste capítulo. Na seção 6.8.2 daremos a descrição dos algoritmos do processo de geração de celas, geração do slot "Restrições" do frame da rota e determinação da rota. Por último, serão descritos os procedimentos a serem seguidos no caso de alterações no Sistema, especificamente no caso de adição de novas restrições, funções de avaliação e atributos.

Encontra-se em anexo uma listagem completa com o código completo desenvolvido para cada um dos módulos do Sistema de Planejamento de Rotas. O equipamento usado foi um micro-computador do tipo IBM/PC-XT com 512K de memória RAM e clock de 4,77 MZ, a linguagem utilizada para a implementação foi uma versão do LISP para micro-computadores (MULISP-86 da Microsoft).

O código do Sistema de Planejamento de Rotas aqui desenvolvido foi sub-dividido fisicamente nos seguintes módulos em LISP :

EXTENDL.LSP --> Biblioteca de funções de utilidade geral (2,5 k) que constituem uma extensão das funções oferecidas pelo Mulisp-86.

GRAPH.LSP (6 k) ---> Biblioteca de funções que permitem a criação de gráficos utilizando o Mulisp-86. É formado por algumas funções da biblioteca GRAPHICS.LSP fornecida pelo fabricante mais algumas funções específicas desenvolvidas para o Sistema.

FILEIO.LSP (1 k) ---> Biblioteca de funções que permitem a manipulação de arquivos pelo Sistema.

MENUP.LSP (10,5 k) ---> Formado por funções de chamada do Menu principal e de menus de acesso às rotinas de entrada de dados de obstáculos e limites do layout.

GERACELL.LSP (20 k) ---> Contêm as funções utilizadas pelo Sistema para a geração de celas obstáculo-adaptativas.

INFERENC.LSP (9 k) ---> Composto pelas funções que constituem o módulo de inferência desenvolvido para o Sistema.

ROUTER1.LSP (18,5 k) ---> Contêm as funções utilizadas pelo Sistema para a entrada do contexto do problema.

ROUTER2.LSP (3,5 k) ---> Contêm as funções necessárias à geração de frames e sub-frames da rota a ser planejada.

ROUTER3.LSP (24,5 k) ---> Contêm as funções de determinação da rota, avaliação de restrições, funções específicas de verificação de restrições,

interação com o usuário e emissão de diagnósticos/resultados.

6.8.1 Estruturas de dados.

A representação de frames como dos seus slots, composição de funções de avaliação, celas, obstáculos, areas da configuração, clausulas e regras foi feita através de listas ou multi-listas aproveitando a facilidade da linguagem LISP de manipular este tipo de estrutura de dados. A representação das celas geradas no processo dde geração de celas obstáculo-adaptativas foi abordada no Capitulo III. A seguir mostraremos a representação usada em nossa implementação para áreas da configuração e frames; no caso específico do algoritmo de busca do caminho, a representação usada para celas atingidas e/ou expandidas e das variáveis sujeitas a restrições associadas a cada cela atingida e avaliada pelo sub-módulo de avaliação de restrições (Ver algoritmo de busca do caminho na seção 6.8.4).

AREAS DA CONFIGURAÇAO.

(Rótulo da área (lista de rótulos de celas que compõem a área))

Ex.: (A1 (3 5 6 7 9)) Indica que o rótulo da área é A1 e esta é formada pelas celas 3,5,6,7 e 9.

FRAMES.

(("ROTULO",rótulo da rota)
("TIPO ",(lista de clausulas na forma de listas
binárias que descrevem as propriedades da
rota a ser determinada))
("PAI-DE",(lista de rótulos de sub-frames dos quais o
frame é "pai"))
("PARTE-DE",(rótulo do frame "pai" do frame
corrente))
("INICIAL",(rótulo da cela inicial da rota))
("OBJETIVO",(rótulo da cela objetivo))
("VIA-POINTS",(rótulos das celas que constituem pontos
pelos quais a rota deve obrigatoriamente
passar))
("AREAS-DE-PREFERENCIA",(Rótulos das celas que se
encontram em áreas de
preferência da rota))
("RESTRICOES",(lista de restrições da rota))
("RESTRIÇAO1",(lista de atributos das clausulas que
tem "RESTRIÇAO1" como predicado e
entre seus atributos o rótulo da rota a
ser determinada)).

"
"
"
("RESTRIÇAOn",(lista de atributos das clausulas que
tem "RESTRIÇAOn" como predicado e
entre seus atributos o rótulo da rota a
ser determinada)).
("FUNCAO-GLOBAL-DE-AVALIACAO",(representação da função
global de avaliação da

rota))

("SOLUÇAO",(lista de celas que formam a rota

solução)))

Exemplo de frame da rota gerado pelo sistema :

```
((ROTULO R1)
 (TIPO ((SENSIVEL-AO-CALOR R1)(SENSIVEL-A-UMIDADE)))
 (PAI-DE NIL)
 (PARTE-DE R1)
 (INICIAL 2)
 (OBJETIVO 10)
 (VIA-POINTS (2 10))
 (AREAS-DE-PREFERENCIA (15 4 3))
 (RESTRIÇOES (DISTANCIA-MINIMA SEGMENTO-MINIMO))
 (DISTANCIA-MINIMA ((R1 01 3)
                    (R1 03 5)
                    (R1 DEFAULT 1)))
 (SEGMENTO-MINIMO 10)
 (FUNÇAO-GLOBAL-DE-AVALIAÇAO (((1 DISTANCIA-MANHATTAM)(1
                              ESTIMATIVA-DISTANCIA-MANHATTAM)))
 (SOLUÇAO (10 7 5 3 2))    )
```

Significando :

- O rôtulo da rota é R1.
- O tipo de rota é : Sensível ao calor e sensível à umidade.
- O frame é parte de ele mesmo apenas (não possui um frame "pai").
- O ponto de partida da rota é o centro de gravidade da cela 2.
- O ponto de chegada da rota é o centro de gravidade da cela

10.

- A rota possui como "via-points" apenas as celas 2 e 10.
- As celas 15, 4 e 3 encontram-se em áreas de preferência da rota.
- As restrições aplicadas à rota são "distancia minima" e "segmento mínimo".
- A distância mínima a ser mantida é de 3 unidades em relação ao obstáculo 01, 5 unidades em relação a 03 e de 1 para qualquer outro obstáculo (distância "default").
- Qualquer segmento de rota não pode ser menor que 10 unidades.
- A função de avaliação é formada por um único componente que por sua vez é formado pelas funções de avaliação da distância manhattam e de estimativa de distância manhattam com coeficientes de ponderação 1 e 1 respectivamente.
- A rota solução é formada de trás para frente pelas celas 10, 7, 5, 3 e 2.

REPRESENTAÇAO DE CELAS PELO MODULO DE BUSCA DO CAMINHO.

Durante a busca da rota, é atingido um sub-conjunto das celas da configuração. O módulo de busca do caminho utiliza uma representação separada das celas atingidas e os seus custos associados. Esta representação é idêntica tanto para celas expandidas como não expandidas.

(Rótulo da cela (Rótulo da cela "pai"
((vf1,vh1),...,(vfn,vhn)) custo total
))

vfn -> custo de atingir a cela determinado pela função de

medida de performance fn.

vhn -> estimativa do custo para atingir a cela objetivo efetuada pela função de estimação hn.

custo total -> soma dos valores parciais determinados por cada componente da função global de avaliação.

A geração da representação interna da função global de avaliação foi abordada na seção 6.3 deste capitulo. O módulo de busca do caminho utiliza paralelamente uma lista de variáveis sujeitas a restrições para cada cela que representam informações locais obtidas durante a busca pelo algoritmo de busca do caminho, mais especificamente, pelo sub-modulo de avaliação de restrições (ver algoritmo na seção 6.5). A estrutura de cada elemento desta lista é a seguinte :

(Rótulo da cela ((Rótulo da restrição 1 (clausula 1, clausula 2, ...,clausula n))
(Rótulo da restrição 2 (clausula 1, clausula 2, ...,clausula n)) ...
(Rótulo da restrição n (clausula 1, clausula 2, ...,clausula n))))

Exemplo :

(5 ((DISTANCIA-MINIMA)(SEGMENTO-MINIMO ((DIRECAO-ANTERIOR NORTE)(SEGMENTO 5)(CGC-ANT (20 15))))))

Significando :

A cela de rótulo 5 atingida durante a busca satisfaz a restrição de distância mínima cuja avaliação dispensa o armazenamento de variáveis sujeitas a restrições

(não possui clausulas associadas). É satisfeita também a restrição de segmento mínimo sendo armazenadas a direção atual (Norte) que será considerada por uma cela vizinha como a direção anterior da rota, o segmento de rota que tem na cela 5 um dos seus vértices tem comprimento igual a 5 e o centro de gravidade da cela está localizado nas coordenadas (20;15).

6.3.2 Algoritmo de Geração de Celas.

Etapa 1. (Definir o conjunto de coordenadas dos obstáculos e limites do layout)

- Entrar com as coordenadas dos vértices superior esquerdo, superior direito, inferior direito e inferior esquerdo do layout da configuração considerando o vertice inferior esquerdo como a origem do Sistema de Coordenadas (X-Y).

- Para cada obstáculo da configuração faça :
  Entrar com as coordenadas dos vértices superior esquerdo, superior direito, inferior direito e inferior esquerdo do obstáculo.

Etapa 2. (Defina os blocos primitivos na base de conjuntos de coordenadas.)

- Para cada obstáculo da configuração faça :

- Prolongar as arestas dos obstáculos até intersectar os limites Sul e Oeste do layout. Adicionar as coordenadas dos pontos de intersecção com o limite Sul obtidos à lista de coordenadas que representam intersecções com

este limite considerado como o Eixo X, caso estas coordenadas não façam parte desta lista. Proceder da mesma maneira com as interseções com o limite Oeste considerado por convenção como o Eixo Y.

- Classificar ascendentemente as listas de interseções com os Eixos X e Y, obtidas no passo anterior.

- Para cada elemento y da lista de interseções com Y faça:

    Para cada elemento x da lista de interseções com X faça:

        A partir de x e y obter um bloco retangular cujo vértice inferior direito seja x e seu vértice superior esquerdo, y.

        Atribuir um número como rótulo ao bloco gerado igual ao rótulo do bloco anterior mais um.

        Adicionar a lista de coordenadas à lista de atributos do bloco (cela).

        Faça y igual ao vértice superior direito do bloco gerado.

    fim-para.

    Faça o conjunto de vértices superiores esquerdo do conjunto de blocos obtido na etapa anterior, classificados ascendentemente, a nova lista de interseções com o Eixo X.

Fim-para.

- (Geração de "vizinhos" de uma cela).

Seja m o número de celas em uma fileira horizontal obtida na configuração obtida.

Para cada bloco (cela) obtido na etapa anterior faça :

Faça L igual ao rótulo do bloco.

Se a aresta norte da cela não é coincidente com o limite norte do layout, L + m é o rótulo do vizinho norte da cela, senão, o vizinho é o limite norte do layout.

Se a aresta leste da cela não é coincidente com o limite leste do layout, L + 1 é o rótulo do vizinho leste da cela, senão, o vizinho é o limite leste do layout.

Se a aresta sul da cela não é coincidente com o limite sul do layout, L - m é o rótulo do vizinho sul da cela, senão, o vizinho é o limite sul do layout.

Se a aresta oeste da cela não é coincidente com o limite oeste do layout, L - 1 é o rótulo do vizinho oeste da cela, senão, o vizinho é o limite oeste do layout.

Adicionar a lista de vizinhos obtida à lista de atributos da cela.

Fim-para.

- (Geração de "visões" da cela)

Para cada cela C da lista de celas geradas faça :

Para cada uma das quatro direções possíveis para a cela faça:

Seja a distância mínima detetada D igual a 0.

Seja a variável de detecção S igual a Falso (F).

Para cada obstáculo O da lista de obstáculos faça:

Se a distância de C para O na direção avaliada é menor ou igual a D então faça :

D igual à distância achada.

Armazene o rótulo do obstáculo.

Se S falso (F), faze-la igual a Verdadeiro (T).

Fim-se.

Fim-para.

Se a variável de detecção S é falos (F) então faça a visão da cela na direção corrente igual ao tipo de limite do layout naquela direção. (Norte, Sul, Leste ou Oeste).

Fim-para.

Adicione a lista de visões obtida aos atributos da cela.

Fim-para.

Etapa 3. (Teste)

- Para cada cela c_1 da lista de celas faça :

Se c_1 encontra-se completamente contida em

algum obstáculo da configuração então elimine c_i da lista de celas, e atualize a lista de "vizinhos" das celas vizinhas a c_i. (O obstáculo detectado é agora um dos "vizinhos" destas celas e não mais c_i).

Fim-para.

Etapa 4. (Combinação).

- Seja G um "flag" de detecção global de combinações igual a "Verdadeiro" (T).

- Enquanto G for verdadeiro faça :

Seja L um "flag" de detecção de combinação na iteração igual a "Falso" (F).

Para cada cela c_i da lista de celas faça :

Para cada uma das celas vizinhas c_v de c_i faça:

Se as "visões" de c_i forem idênticas às de c_v faça:

Combinar c_i e c_v para obter uma nova cela c_i.

Eliminar c_v da lista de celas.

Atualizar os atributos de c_i e dos "vizinhos" de c_v na lista de celas.

Faça o "flag" de detecção igual a "verdadeiro".

Fim-se.

Fim-para.

Fim-para.

Se L for "falso" então faça G igual a "falso".

Fim-enquanto.

6.8.3 Algoritmo de Geração do slot "Restrições" do frame da rota.

- Seja a variável RESTRICOES-DO-SLOT igual à lista nula.

- (Recuperação da lista de restrições aceitas pelo Sistema) Recuperar todas as instanciações do predicado "Restrição" e atribuir o resultado à variável RESTRICOES.

- (Formação do conteúdo do slot)
Enquanto RESTRICOES for diferente de nulo faça :

  - Seja ATM o primeiro elemento do conteúdo corrente de RESTRICOES.

  - Se ATM não é predicado de nenhuma clausula na Base de Conhecimento, então exclua ATM da lista RESTRICOES.
  Senão faça :
    - Seja INSTANCIACOES igual a lista de clausulas que possuem a restrição ATM como predicado e como um dos seus argumentos o rótulo da rota que está sendo planejada.
    - Se INSTANCIACOES for nulo, exclua ATM da lista RESTRICOES senão adicionar ATM à lista RESTRICOES-DO-SLOT e fazer a lista RESTRICOES igual a ela mesma menos o seu primeiro elemento (ATM).

  Fim-Se.

Fim-Enquanto.

- (Retorno da configuração final do slot "Restricoes")

Criar e retornar a lista que contêm o símbolo "Restricoes" como primeiro elemento e a lista RESTRICOES-DO-SLOT como segundo elemento.

6.8.4 Algoritmo de Determinação da Rota.

Etapa 1 (Inicialização)

- Coloque a cela inicial na lista WF de celas atingidas e que não foram ainda expandidas.
- Seja LST-VARIAVEIS a lista das variáveis sujeitas a restrições das celas atingidas e/ou expandidas.

Etapa 2 (Determinação da rota)

- Enquanto WF não for vazia faça :
  - Ache a cela de menor custo E em WF.
  - Excluir E de WF e inclui-la em WB (Lista de celas expandidas).
  - Se a cela E for a cela objetivo, ativar o procedimento de recuperação da rota e pare.
  - Enquanto não forem atingidas todas as celas vizinhas de E faça:
    - Selecione uma cela vizinha N.
    - Invocar o módulo de avaliação de restrições (que compreende também o módulo de interação com o projetista) para testar a viabilidade de N e cálculo de variáveis sujeitas a restrições.
    - Se N foi considerada viavel então faça :
      - Calcule o custo de atingir N via E e incluir este custo na lista de atributos de N.

- Se N pertence a WF e se o custo calculado é menor que o custo anterior então faça :
  - Exclua o valor de N em WF.
  - Exclua os valores das variáveis sujeitas a restrições de N em LST-VARIAVEIS.
  - Inclua o novo valor de N em WF.
  - Inclua os novos valores das variáveis sujeitas a restrições de N em LST-VARIAVEIS.

  Senão faça :
  - Se N pertence a WB e se o custo calculado é menor que o custo anterior então faça :
    - Exclua o valor de N em WB.
    - Exclua os valores das variáveis sujeitas a restrições de N em LST-VARIAVEIS.
    - Inclua o novo valor de N em WF.
    - Inclua os novos valores das variáveis sujeitas a restrições de N em LST-VARIAVEIS.

    Senão faça :
    - Se N não pertence nem a WF nem a WB então faça :
      - Adicione N a WF.
      - Adicione os valores das variáveis sujeitas a restrições a LST-VARIAVEIS.

      Fim-Se.

    Fim-Se.

  Fim-Se.

Fim-Se.

Fim-Enquanto.

Fim-Enquanto.

- Enviar Mensagem de erro (Não é possivel determinar a rota) e pare.

6.8.5 Adição de novas restrições, funções de avaliação e atributos.

Uma meta ideal a ser atingida por programas de I.A. é a de permitir a soluçao de um problema apenas pela descrição deste sem precisar de alterações da estrutura do programa em si. O esquema de reduçao do conhecimento do sistema permite com que um número relativamente grande de fatos que descrevem o contexto do problema seja reduzido através de inferência a um número limitado de informações denominado de conhecimento primitivo, que é o realmente utilizado pelo sistema na busca da(s) soluçao(es).

A especificação de funções de avaliação segue no entanto um esquema tradicional que porém permite uma flexibilidade na atribuição de pesos relativos aos componentes da função global de avaliação e na ponderação das funções g e h de cada componente. Apesar de que o esquema de reduçao de conhecimento permite com que uma quantidade relativamente grande e variada de conhecimento seja aceita pelo sistema, é possivel que em alguns casos seja necessário adicionar novos procedimentos para manipulação de novos tipos de conhecimento primitivo assim como novas funções no caso de adição de novos indices de performance. Algumas normas para adição de novos atributos de rotas, obstáculos, áreas e funções de avaliação na base

de conhecimento serão aqui também apresentadas.

ADIÇAO DE RESTRIÇOES.

Para cada restrição adicionada deve ser especificado na base de dados original (FATOS.MEM), as clausulas que estabelecem que um determinado predicado é o nome de uma restrição e se esta restrição é de natureza rígida ou flexivel.

Ex.: (RESTRIÇAO DISTANCIA-MINIMA)
"Distância mínima é o nome de uma restrição aceita pelo Sistema"

(RIGIDA DISTANCIA-MINIMA)
"A restrição distância mínima é do tipo rígido"

(FLEXIVEL VIZINHANCA-NAO-RECOMENDADA)
"A restrição vizinhança não recomendada é do tipo flexivel".

Isto é devido a que o sistema procura na base de conhecimento as restrições possiveis de serem aplicadas ao problema e através de interação com o projetista obtêm a informação de quais destas restrições serão de fato aplicadas na determinação da rota. Como o sistema não possui uma interface de linguagem de natural que traduza a informação contida na base de conhecimento para a linguagem do usuário, para cada restrição feita pelo sistema é criada uma função especifica que envia pedidos de confirmação ao projetista e obtêm deste a resposta de se uma determinada restrição será aplicada ou não à rota. Por outro lado, caso o projetista não confirme a aplicação da restrição por

desconhecimento ou não, esta confirmação poderá ser obtida diretamente da base de conhecimento por inferência caso as informações disponíveis permitam a instanciação das regras específicas. O rótulo da função de interação com o usuario deve ser composta pela concatenação da seqüência de caracteres "INF-" mais o nome da restrição tal como aparece na base de conhecimento. Outras normas a serem seguidas são as seguintes :

Argumentos de entrada :

LBL -> Número de seqüência da rota que está sendo determinada.

Ex.: 2,3,5,etc.

Processamento realizado :

- Adição da lista simbolizando a clausula que determina a aplicação da restrição na determinação da rota na lista de rótulo ATTRIB-RESTR que contêm as clausulas que representam o conhecimento sobre restrições obtido através de interação com o projetista.
- Adição do rótulo da restrição na lista de rótulo RESTR-ROTA que contêm os rótulos das restrições que serão aplicadas à rota obtidos através de interação com o projetista.

Argumentos retornados : Não tem.

É criada ainda a função especifica de avaliação da restrição que tem como rótulo o nome da restrição tal como aparece na base de conhecimento. As normas para confecção da restrição são as seguintes:

Argumentos de Entrada :

FRM-IMG -> Frame da rota.

E-VARS -> Lista de variáveis sujeitas a restrições da cela "pai" (Ver seção 6.6.1).

N-CEL -> Lista de atributos da cela na lista de celas obtida no processo de geração de celas.

E-CEL -> Lista de atributos da cela que está sendo avaliada na lista de celas.

Processamento realizado :

Verificação de se o segmento de rota contido na cela satisfaz a restrição.

Argumento retornado :

Lista que obedece a seguinte estrutura :

(Resultado da avaliação,(Rótulo da restrição,
(Lista de variáveis sujeitas a restrições))

Resultado da avaliação ->"SUCCESS", no caso de satisfação da restrição.
"FAIL", no caso de fracasso.

Rótulo da restrição -> Nome da restrição tal como aparece na base de conhecimento.

Lista de variáveis sujeitas a restrições -> Criada quando as informações obtidas na avaliação serão de utilidade para a avaliação de celas sucessoras (Ver estrutura na seção 6.6.1).

ADIÇAO DE HEURISTICAS.

Aqui também é necessária uma função de interação com o projetista que indagará sobre a aplicabilidade ou não da heurística ao problema. Denominamos aqui de heurística um determinado conhecimento não restritivo sobre o problema que orientará a busca da solução. Esta orientação se viabiliza no uso conjugado de funções especificas com o algoritmo de busca do caminho. O procedimento seguido para aceitação de heurísticas é o mesmo que no caso das restrições, cabendo no entanto a nós definir como este conhecimento será de utilidade na busca da solução e a concepção das funções adicionais que se façam necessárias. As clausulas geradas na função de interação com o projetista são adicionadas à lista ATTRIB-HEUR.

Em nossa implementação "áreas de preferência" é a denominação dada a um tipo de conhecimento heurístico que serve para identificar as áreas que são preferidas para a passagem da rota, o aproveitamento desta informação se deu pela criação de uma função de avaliação de rótulo "AREAS-DE-PREFERENCIA" que atribue um custo menor às celas que encontram-se dentro de áreas de preferência.

ADIÇAO DE FUNÇOES DE AVALIAÇAO.

Na rotina de entrada da função global de avaliação são entrados os códigos que identificam os componentes da função, assim por exemplo, para a distância Manhattam o código é DM. Esta informação é obtida pelo Sistema da base de conhecimento original a partir de clausulas do tipo :

(EQUIVALENCIA-FUNÇAO-MANIPULAVEL DM DISTANCIA-MANHATTAM)

Assim para cada nova função de medida de performance ou estimação, deve ser adicionada uma clausula neste formato. A função de avaliação deve ter rótulo igual à tradução do código desta na clausula respectiva na base de conhecimento (Em nosso exemplo, "DISTANCIA-MANHATTAM"). As normas para confecção da função de avaliação são as seguintes :

Argumentos de entrada :

CUSTO-P -> componente da lista global de custos da cela "pai" associada a esta cela ao qual a função pertence.

FILHO -> lista de atributos na lista de celas da cela cujo custo está se querendo determinar

PAI-B -> lista de atributos na lista de celas da cela "pai" da cela cujo custo está se querendo determinar.

FRM-W -> Frame da rota.

Processamento realizado :

Determinação do custo de se atingir a cela, no caso de funções de medida de performance ou estimativa do custo de se atingir a solução no caso de funções de estimação.

Argumentos retornados :

Valor representando o custo de se atingir a cela, no caso de funções de medida de performance ou valor da estimativa de se atingir a solução a partir da cela, no caso de funções de estimação.

ADIÇAO DE ATRIBUTOS MANIPULAVEIS PELO SISTEMA.

Embora a única recomendação que deveria ser feita em relação a entrada de clausulas no sistema é a que existissem regras aptas a manipular estas clausulas, em nosso sistema precisamos de um cuidado adicional em relação a clausulas que descrevem atributos para rotas, obstáculos e áreas da configuração. Como o módulo de entrada de contexto da rota pesquisa na base de dados original do sistema quais atributos aplicáveis em cada caso, é preciso dar os meios para o sistema fazer esta distinção. Assim para cada atributo possivel para rotas tem que constar na base de conhecimento a clausula (DESCRITOR-DE-ROTA nome-do-atributo-da-rota).

Ex.: (DESCRITOR-DE-ROTA SENSIVEL-AO-CALOR)

"Sensivel ao calor é um predicado possivel para a rota"

No caso de atributos para obstáculos o formato é o seguinte :

(DESCRITOR-DE-OBSTACULO nome-do-atributo-do-obstáculo)

Finalmente no caso de atributos para áreas :

(DESCRITOR-DE-AREA nome-do-atributo-da-área).

CAPITULO VII - RESULTADOS EXPERIMENTAIS E CONCLUSÕES

Neste capitulo serão apresentados os resultados de alguns testes realizados com o Sistema de Planejamento de Rotas e apresentadas as conclusões finais da tese. No inicio da apresentação dos resultados, serão mostradas as regras e fatos que compõem a base de conhecimento inicial do Sistema usada em nossos testes. Foram realizados 7 testes de planejamento de rotas. Em cada teste foram determinadas 4 rotas, sempre com os mesmos pontos inicial e objetivo.

7.1 Resultados Experimentais.

BASE DE CONHECIMENTO INICIAL.

Regra P1 : "Se L é sensivel à gravidade e a direção norte representa uma direção para cima, então a direção norte é uma direção proibida para L".

Regra P2 : "Se L é sensivel ao calor, A é quente e A é uma área, então A é uma área proibida para L"

Regra P3 : "Se L é sensivel ao calor, O é quente e O é um obstáculo, então a distância mínima de L para O deve ser igual a 10.

Regra P4 : "Se L é uma rota então a distância mínima assumida de L para qualquer rota é igual a 1".

Regra P5 : "Se L é sensivel ao calor, O é muito quente e O é um obstáculo, então a distância mínima de L para O deve ser igual a 30".

Regra P6 : "Se L é sensivel ao calor, A é pouco quente e A é uma área, então A é uma área não recomendada

para L".

Regra P7 : "Se L é sensível ao calor, A é muito quente e A é uma área, então A é uma área proibida para L".

Regra P8 : "Se L é sensível a baixa temperatura, O é frio e O é um obstáculo, então O é uma vizinhança proibida para L".

Regra P9 : "Se L é sensível a baixa temperatura, O é muito frio e O é um obstáculo, então a distância mínima de L para O deve ser igual a 5".

Regra P10: "Se L é sensível a baixa temperatura, O é pouco frio e O é um obstáculo, então O é uma vizinhança não recomendada para L".

Regra P11: "Se L é sensível à baixa temperatura, A é muito fria e A é uma área, então A é uma área proibida para L".

Regra P12: "Se L é sensível à baixa temperatura, A é pouco fria e A é uma área, então A é uma área não recomendada para L".

Regra P13: "Se L é sensível à corrosão, O é umido e O é um obstáculo, então a distância mínima de L para O deve ser igual a 2".

Regra P14: "Se L é sensível à corrosão, A é umida e A é uma área, então A é uma área proibida para L".

Regra P15: "Se L é uma rota, O é sujeito a incerteza e O é um obstáculo, então a distância mínima de L para O deve ser igual a 2".

Regra P16: "Se L é foto-química, O é emissor de luz e O é um obstáculo, então a distância mínima de L para O deve ser igual a 3".

Regra P17: "Se L é foto-química, A é iluminada e A é uma

área, então A é uma área proibida para L".

Regra P18: "Se L é favorável a iluminação, A é iluminada e A é uma área, então A é uma área de preferência para L".

Regra P19: "Se L é favorável a ventilação, A é ventilada e A é uma área, então A é uma área de preferência para L".

Regra P20: "Se L é sensível ao magnetismo, O é eletro-magnético, então a distância mínima de L para O deve ser igual a 4".

Regra P21: "Se L é sensível ao magnetismo, A é eletro-magnética e A é uma área, então A é uma área proibida para L".

Regra P22: "Se as rotas da configuração possuem a mesma cota de elevação e L é uma rota com dimensões, então não é possível o cruzamento de L com outra rota".

Regra P23: "Se L é sensível a gases, A está contaminada de gases nocivos e A é uma área, então A é uma área proibida para L".

Regra P24: "Se L é sensível a ruidos, A está sujeita a ruidos elevados e A é uma área, então A é uma área não recomendada".

Regra P25: "Se L é sensível a baixa temperatura, A é fria e A é uma área, então A é uma área proibida para L"

"Quente é um descritor de obstáculos".

"Frio é um descritor de obstáculos".

"Qualquer-tipo é um atributo descritor de obstáculos".

"Muito-quente é um atributo descritor de obstáculos".

"Pouco-quente é um atributo descritor de obstáculos".

"Muito-frio é um atributo descritor de obstáculos".

"Pouco-frio é um atributo descritor de obstáculos".

"Umido é um atributo descritor de obstáculos".

"Eletro-magnético é um atributo descritor de obstáculos".

"Emanador-de-ar é um atributo descritor de obstáculos".

"Sujeito-a-incerteza é um atributo descritor de obstáculos".

"Emissor-de-luz é um atributo descritor de obstáculos".

"Emanador-de-gases-nocivos é um atributo descritor de obstáculos".

"Quente é um atributo descritor de área".

"Fria é um descritor de área".

"Muito-fria é um atributo descritor de área".

"Pouco-fria é um atributo descritor de área".

"Quente é um atributo descritor de área".

"Umida é um atributo descritor de área".

"Eletro-magnética é um atributo descritor de área".

"Ventilada é um atributo descritor de área".

"Iluminada é um atributo descritor de área".

"Contaminada-de-gases-nocivos é um atributo descritor de área".

"Sujeita-a-ruidos-elevados é um atributo descritor de área".

"Qualquer-tipo é um atributo descritor de rota".

"Sensivel-ao-calor é um atributo descritor de rota".

"Sensivel-a-baixa-temperatura é um atributo descritor de rota".

"Sensivel-a-varios-niveis-de-ruidos é um atributo descritor de rota".

"Foto-química é um atributo descritor de rota".

"Favorável-a-ventilação é um atributo descritor de rota".

"Favorável-a-iluminação é um atributo descritor de rota".

"Sensível-a-gases é um atributo descritor de rota".

"Sensível-a-corrosão é um atributo descritor de rota".

"Sensível-a-magnetismo é um atributo descritor de rota".

"Distância-mínima é o nome de uma restrição de rota".

"Distância-mínima-recomendada é o nome de uma restrição de rota".

"Direção-proibida é o nome de uma restrição de rota".

"Segmento-mínimo é o nome de uma restrição de rota".

"Vizinhança-proibida é o nome de uma restrição de rota".

"Vizinhança-não-recomendada é o nome de uma restrição de rota".

"Área-proibida é o nome de uma restrição de rota".

"Cruzamento-proibido é o nome de uma restrição de rota".

"Coincidência-proibida é o nome de uma restrição de rota".

"Largura-mínima-da-rota é o nome de uma restrição de rota".

"Áreas-de-preferencia é o nome de uma heurística do problema".

"Distância-mínima é uma restrição do tipo rígido".

"Vizinhança-proibida é a denominação de uma restrição do tipo rígido".

"Área-proibida é a denominação de uma restrição do tipo rígido".

"Direção-proibida é a denominação de uma restrição do tipo rígido".

"Segmento-mínimo é a denominação de uma restrição do tipo rígido".

"Distância-mínima-recomendada é a denominação de uma restrição do tipo flexível".

"Vizinhança-não-recomendada é a denominação de uma restrição do tipo flexivel".

"Area-não-recomendada é a denominação de uma restrição do tipo flexivel".

"DM é o código que identifica a distância Manhattam como uma medida de performance".

"EDM é o código que identifica a estimativa da distância Manhattam como uma medida de performance".

"NC é o código que identifica o número de cruzamentos da rota como uma medida de performance".

"AP é o código que identifica o número de segmentos em áreas de preferência como uma medida de performance".

"EAP é o código que identifica a estimativa de segmentos em áreas de preferência como uma medida de performance".

Foi utilizada nos testes uma configuração com 20 obstáculos com as seguintes coordenadas de vértices V.S.E. (Vértice Superior Esquerdo), V.S.D. (Vértice Superior Direito), V.I.D. (Vértice Inferior Direito), V.I.E. (Vértice Inferior Esquerdo).:

Limites do layout :

V.S.E. -> (0;140)

V.S.D. -> (260;140)

V.I.D. -> (260;0)

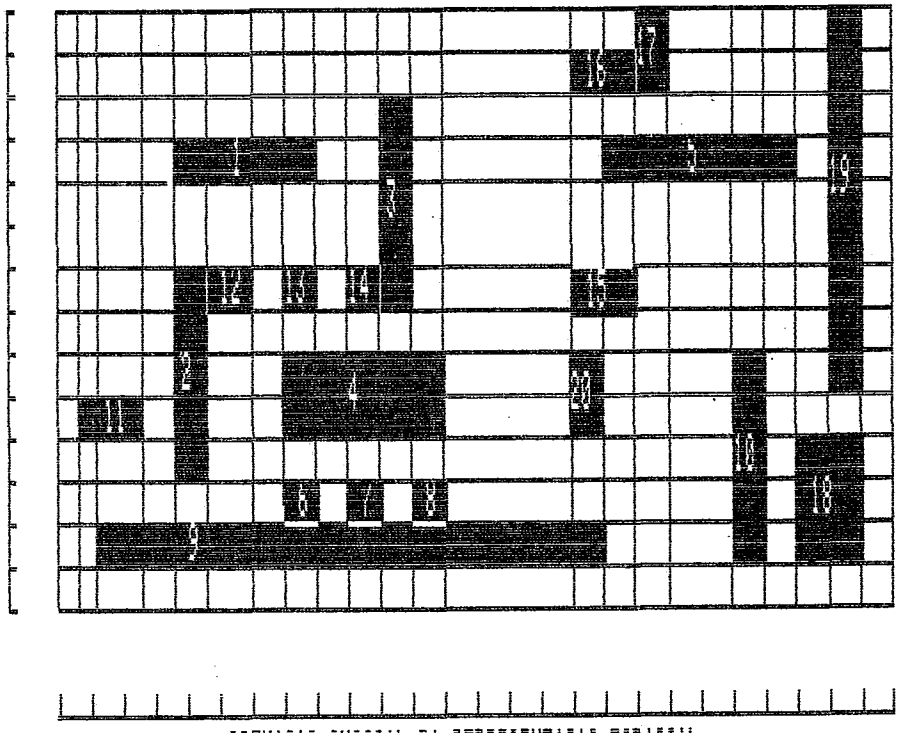
V.I.E. -> (0;0)

Coordenadas dos obstáculos :

Rotulo	V.S.E.	V.S.D.	V.I.D.	V.I.E.
01	(35;110)	(80;110)	(80;100)	(35;100)
02	(35;80)	(45;80)	(45;30)	(35;30)

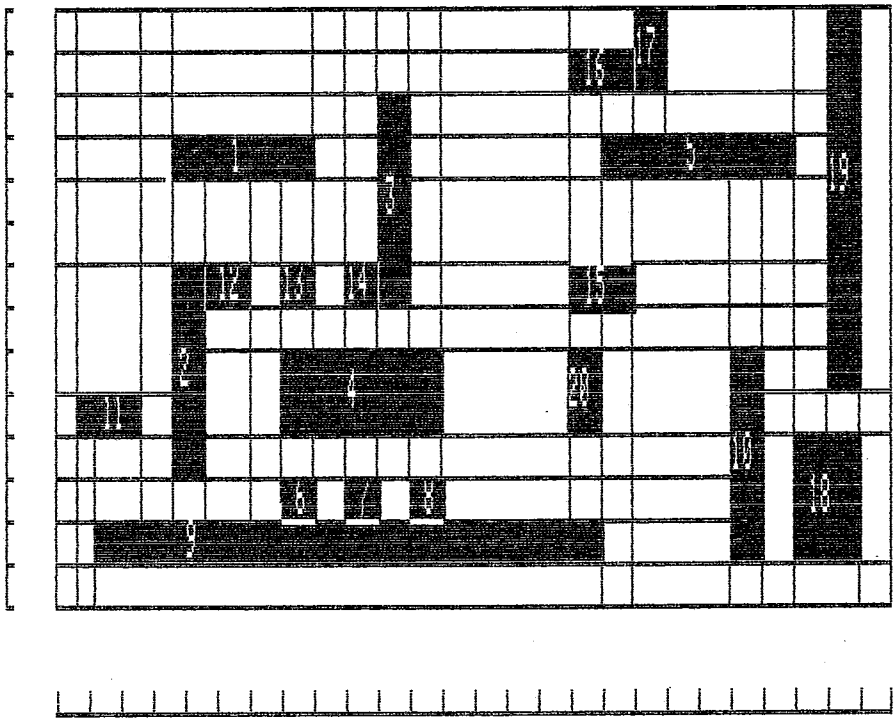
03	(100;120)	(110;120)	(110;70)	(100;70)
04	(70;60)	(120;60)	(120;40)	(70;40)
05	(170;110)	(230;110)	(230;100)	(170;110)
06	(70;30)	(80;30)	(80;20)	(70;20)
07	(90;30)	(100;30)	(100;20)	(90;20)
08	(110;30)	(120;30)	(120;20)	(110;20)
09	(10;20)	(170;20)	(170;10)	(10;10)
010	(210;60)	(220;60)	(220;10)	(210;10)
011	(5;50)	(25;50)	(25;40)	(5;40)
012	(45;80)	(60;80)	(60;70)	(45;70)
013	(70;80)	(80;80)	(80;70)	(70;70)
014	(90;80)	(100;80)	(100;70)	(90;70)
015	(160;80)	(180;80)	(180;70)	(160;70)
016	(160;130)	(180;130)	(180;120)	(160;120)
017	(180;140)	(190;140)	(190;120)	(180;120)
018	(230;140)	(250;40)	(250;10)	(230;10)
019	(240;140)	(250;140)	(250;50)	(240;50)
020	(160;60)	(170;60)	(170;40)	(160;40)

Junto com os gráficos que ilustram os resultados, aprarecem duas escalas, uma vertical e outra horizontal. Cada variação na escala representa 10 unidades. O Módulo de geração de celas gerou numa primeira etapa 286 celas, reduzidas após as etapas de Teste e Combinação para 142 (Figuras VII.1 e VII.2 respectivamente). O tempo total de processamento utilizado para geração das celas foi de 23 minutos. As rotas determinadas pelo sistema são representadas gráficamente como um conjunto de segmentos de traço cheio para as rotas com rótulos pares e com traço intermitente para as rotas com rótulos ímpares.



SITUACAO INICIAL DA REPRESENTACAO ESPACIAL

FIGURA VII.1



SITUACAO DAS CELAS APOS AS ETAPAS DE TESTE E COMBINACAO

FIGURA VII.2

7.1.1) Teste 1.

Os resultados deste teste podem ser vistos gráficamente na figura VII.3.

- Rota 1 (R1).

função de avaliação : F = DM, DM -> distância manhattam percorrida.

fatos adicionados : "R1 é uma rota"
"O espaçamento minimo assumido de R1 para qualquer obstáculo é igual a 1".

regras instanciadas : P4.

ponto inicial	: (20;35)
ponto objetivo	: (165;105)
número de celas da rota	: 19
número de celas expandidas	: 110
número de celas atingidas	: 116
número de celas da configuração	: 142
distância manhattam percorrida	: 237,5
tempo de geração da rota	: 3 minutos.

- Rota 1 (R1).

função de avaliação : F = DM, DM -> distância manhattam percorrida.

fatos adicionados : "R1 é uma rota"
"O espaçamento minimo assumido de R1 para qualquer obstáculo é igual a 1".

regras instanciadas : P4.

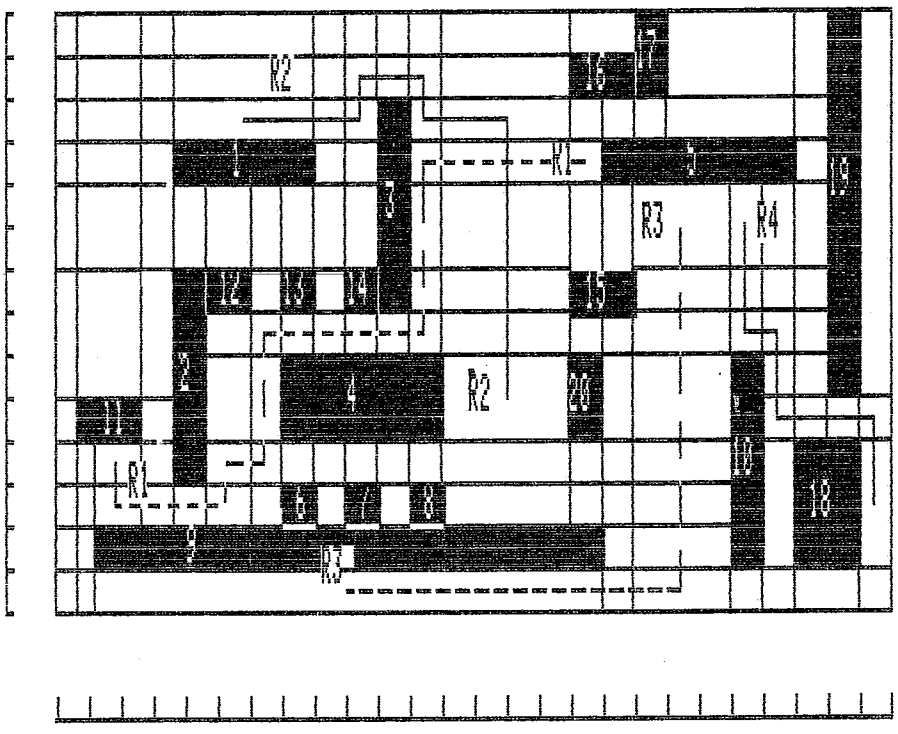


FIGURA VII.3

via-points : (75;65),(115;65)

ponto inicial : (20;35)

ponto objetivo : (165;105)

número de celas da rota : 19

número de celas expandidas : 85

número de celas atingidas : 113

número de celas da configuração : 142

distância manhattam percorrida : 237,5

tempo de geração da rota : 2:30 minutos.

- Rota 2 (R2).

função de avaliação : F = DM, DM -> distância manhattam percorrida.

fatos adicionados : "R2 é uma rota"

"O espaçamento mínimo assumido de R2 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R2 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (50;115)

ponto objetivo : (140;45)

número de celas da rota : 13

número de celas expandidas : 76

número de celas atingidas : 82

número de celas da configuração : 142

distância manhattam percorrida : 167,5

tempo de geração da rota : 2:05 minutos.

- Rota 3 (R3).

função de avaliação : F = DM, DM -> distância manhattam percorrida.

fatos adicionados : "R3 é uma rota"

"O espaçamento minimo assumido de R3 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R3 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial	: (195;90)
ponto objetivo	: (80;5)
número de celas da rota	: 10
número de celas expandidas	: 67
número de celas atingidas	: 71
número de celas da configuração	: 142
distância manhattam percorrida	: 190
tempo de geração da rota	: 1:50 minutos.

- Rota 4 (R4).

função de avaliação : F = DM, DM -> distância manhattam percorrida.

fatos adicionados : "R4 é uma rota"

"O espaçamento minimo assumido de R4 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R4 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (215;90)

ponto objetivo : (255;30)

número de celas da rota : 10

número de celas expandidas : 36

número de celas atingidas : 46

número de celas da configuração : 142

distância manhattam percorrida : 105

tempo de geração da rota : 1:10 minutos.

7.1.2) Teste 2.

Os resultados deste teste podem ser vistos gráficamente na figura VII.4.

- Rota 1 (R1).

função de avaliação : F = DM + EDM,

DM -> distância manhattam percorrida.

EDM-> estimativa da distância manhattam a ser percorrida desde a cela avaliada até a cela objetivo.

fatos adicionados : "R1 é uma rota"

"O espaçamento minimo assumido de R1 para qualquer obstáculo é igual a 1".

regras instanciadas : P4.

ponto inicial : (20;35)

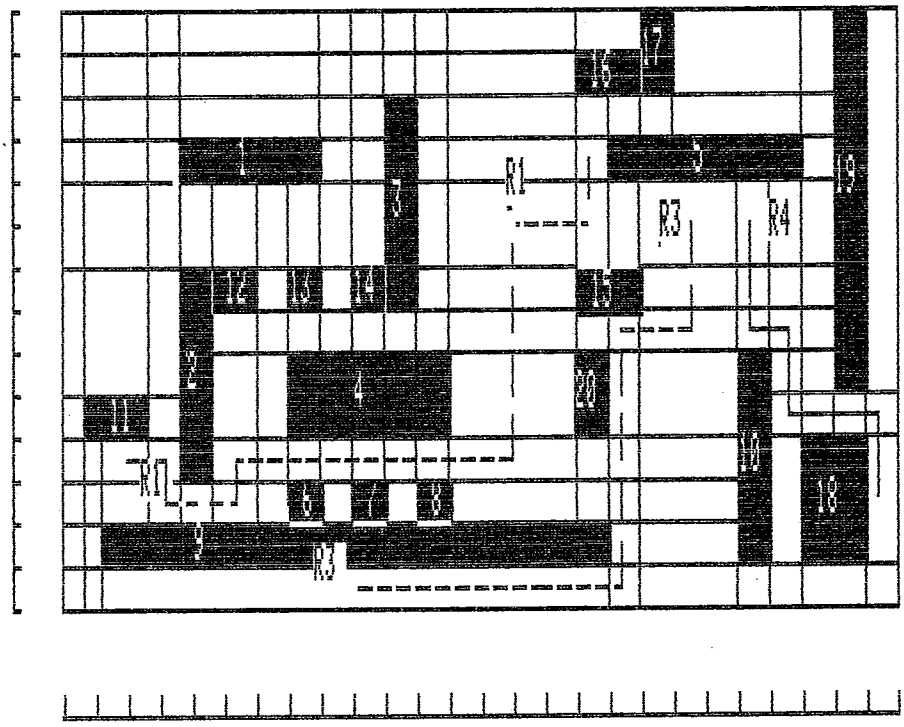


FIGURA VII.4

ponto objetivo : (165;105)

número de celas da rota : 19

número de celas expandidas : 37

número de celas atingidas : 62

número de celas da configuração : 142

distância manhattam percorrida : 237,5

tempo de geração da rota : 1:10 minutos.

- Rota 1 (R1).

função de avaliação : F = DM + EDM,

fatos adicionados : "R1 é uma rota"

"O espaçamento mínimo assumido de R1 para qualquer obstáculo é igual a 1".

regras instanciadas : P4.

via-points : (75;35),(140;150)

ponto inicial : (20;35)

ponto objetivo : (165;105)

número de celas da rota : 19

número de celas expandidas : 22

número de celas atingidas : 37

número de celas da configuração : 142

distância manhattam percorrida : 237,5

tempo de geração da rota : 53 segundos.

- Rota 2 (R2).

função de avaliação : F = DM + EDM

fatos adicionados : "R2 é uma rota"

"O espaçamento mínimo assumido de R2 para qualquer obstáculo é igual a

1".

"A coincidência parcial ou total de R2 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (50;115)

ponto objetivo : (140;45)

solução obtida : "Não é possivel a determinação desta rota".

- Rota 3 (R3).

função de avaliação : F = DM + EDM

fatos adicionados : "R3 é uma rota"

"O espaçamento minimo assumido de R3 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R3 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (195;90)

ponto objetivo : (80;5)

número de celas da rota : 10

número de celas expandidas : 21

número de celas atingidas : 31

número de celas da configuração : 142

distância manhattam percorrida : 190

tempo de geração da rota : 50 segundos.

- Rota 4 (R4).

função de avaliação : F = DM + EDM

fatos adicionados : "R4 é uma rota"

"O espaçamento minimo assumido de R4 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R4 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial	: (215;90)
ponto objetivo	: (255;30)
número de celas da rota	: 10
número de celas expandidas	: 11
número de celas atingidas	: 19
número de celas da configuração	: 142
distância manhattam percorrida	: 105
tempo de geração da rota	: 31 segundos.

7.1.3) Teste 3.

Os resultados deste teste podem ser vistos gráficamente na figura VII.5.

- Rota 1 (R1).

função de avaliação : F = 0,2*DM + 0,8*EDM,

DM -> distância manhattam percorrida.

EDM-> estimativa da distância manhattam a ser percorrida desde a cela

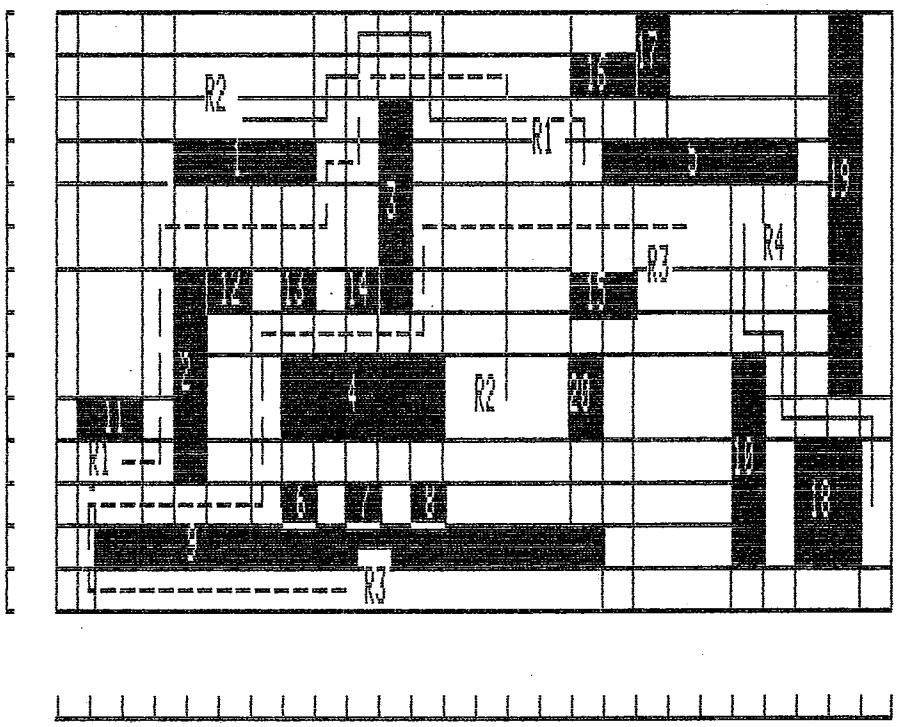


FIGURA VII.5

avaliada até a cela objetivo.

fatos adicionados : "R1 é uma rota"
"O espaçamento minimo assumido de R1 para qualquer obstáculo é igual a 1".

regras instanciadas : P4.

ponto inicial : (20;35)
ponto objetivo : (165;105)
número de celas da rota : 20
número de celas expandidas : 23
número de celas atingidas : 41
número de celas da configuração : 142
distância manhattam percorrida : 257,5
tempo de geração da rota : 38 segundos.

- Rota 2 (R2).

função de avaliação : F = 0,2*DM + 0,8*EDM,
fatos adicionados : "R2 é uma rota"
"O espaçamento minimo assumido de R2 para qualquer obstáculo é igual a 1".
"A coincidência parcial ou total de R2 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (50;115)
ponto objetivo : (140;45)
número de celas da rota : 15

número de celas expandidas : 20

número de celas atingidas : 31

número de celas da configuração : 142

distância manhattam percorrida : 187,5

tempo de geração da rota : 35 segundos.

- Rota 3 (R3).

função de avaliação : F = 0,2*DM + 0,8*EDM

fatos adicionados : "R3 é uma rota"

"O espaçamento mínimo assumido de R3 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R3 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (195;90)

ponto objetivo : (80;5)

número de celas da rota : 23

número de celas expandidas : 46

número de celas atingidas : 56

número de celas da configuração : 142

distância manhattam percorrida : 355

tempo de geração da rota : 1:55 minutos.

- Rota 4 (R4).

função de avaliação : F = 0,2*DM + 0,8*EDM

fatos adicionados : "R4 é uma rota"

"O espaçamento mínimo assumido de R4 para qualquer obstáculo é igual a

1".

"A coincidência parcial ou total de R4 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (215;90)

ponto objetivo : (255;30)

número de celas da rota : 10

número de celas expandidas : 11

número de celas atingidas : 19

número de celas da configuração : 142

distância manhattam percorrida : 105

tempo de geração da rota : 31 segundos.

7.1.4) Teste 4.

Os resultados deste teste podem ser vistos gráficamente na figura VII.6.

- Rota 1 (R1).

função de avaliação : F = 0,35*DM + 0,65*EDM,

DM -> distância manhattam percorrida.

EDM-> estimativa da distância manhattam a ser percorrida desde a cela avaliada até a cela objetivo.

fatos adicionados : "R1 é uma rota"

"O espaçamento mínimo assumido de R1 para qualquer obstáculo é igual a

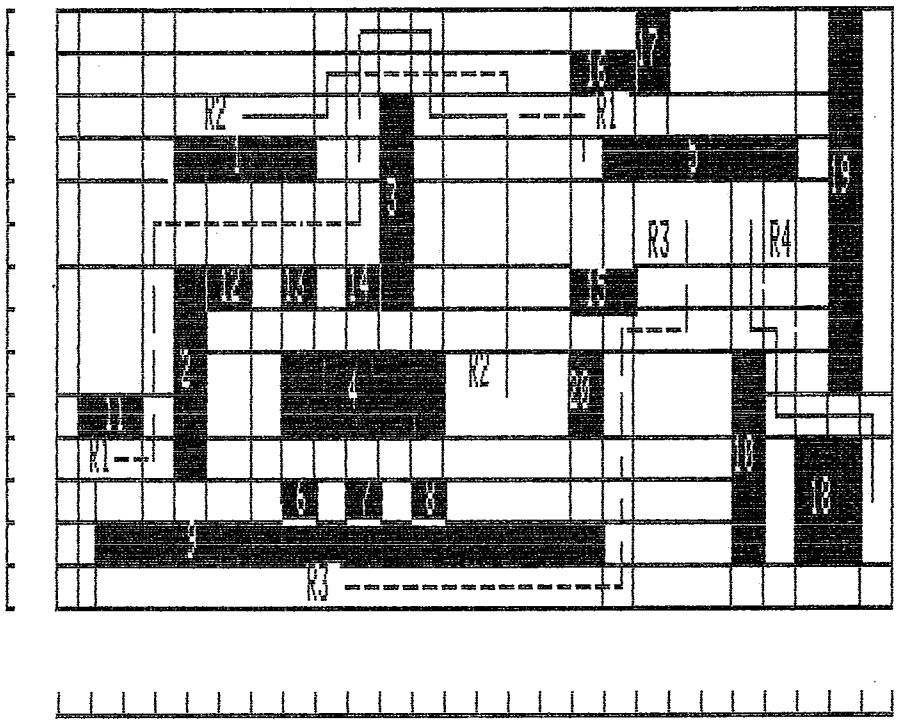


FIGURA VII.6

1".

"A direção Oeste é uma direção proibida para R1".

regras instanciadas : P4.

ponto inicial : (20;35)

ponto objetivo : (165;105)

número de celas da rota : 20

número de celas expandidas : 23

número de celas atingidas : 36

número de celas da configuração : 142

distância manhattam percorrida : 257,5

tempo de geração da rota : 34 segundos.

- Rota 2 (R2).

função de avaliação : F = 0,35*DM + 0,65*EDM,

fatos adicionados : "R2 é uma rota"

"O espaçamento mínimo assumido de R2 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R2 com outra rota da configuração é proibida".

"A direção Oeste é uma direção proibida para R2".

regras instanciadas : P4.

ponto inicial : (50;115)

ponto objetivo : (140;45)

número de celas da rota : 15

número de celas expandidas : 20

número de celas atingidas : 26

número de celas da configuração : 142

distância manhattam percorrida : 187,5

tempo de geração da rota : 28 segundos.

- Rota 3 (R3).

função de avaliação : F = 0,35*DM + 0,65*EDM

fatos adicionados : "R3 é uma rota"

"O espaçamento mínimo assumido de R3 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R3 com outra rota da configuração é proibida".

"A direção Leste é uma direção proibida para R3".

regras instanciadas : P4.

ponto inicial : (195;90)

ponto objetivo : (80;5)

número de celas da rota : 10

número de celas expandidas : 41

número de celas atingidas : 47

número de celas da configuração : 142

distância manhattam percorrida : 190

tempo de geração da rota : 1:46 minutos.

- Rota 4 (R4).

função de avaliação : F = 0,35*DM + 0,65*EDM

fatos adicionados : "R4 é uma rota"

"O espaçamento mínimo assumido de R4

para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R4 com outra rota da configuração é proibida".

"A direção Oeste é uma direção proibida para R4".

regras instanciadas : P4.

ponto inicial	: (215;90)
ponto objetivo	: (255;30)
número de celas da rota	: 10
número de celas expandidas	: 11
número de celas atingidas	: 17
número de celas da configuração	: 142
distância manhattam percorrida	: 105
tempo de geração da rota	: 27 segundos.

7.1.5) Teste 5.

Os resultados deste teste podem ser vistos gráficamente na figura VII.7.

- Rota 1 (R1).

função de avaliação : F = DM + EDM,

DM -> distância manhattam percorrida.

EDM-> estimativa da distância manhattam a ser percorrida desde a cela avaliada até a cela objetivo.

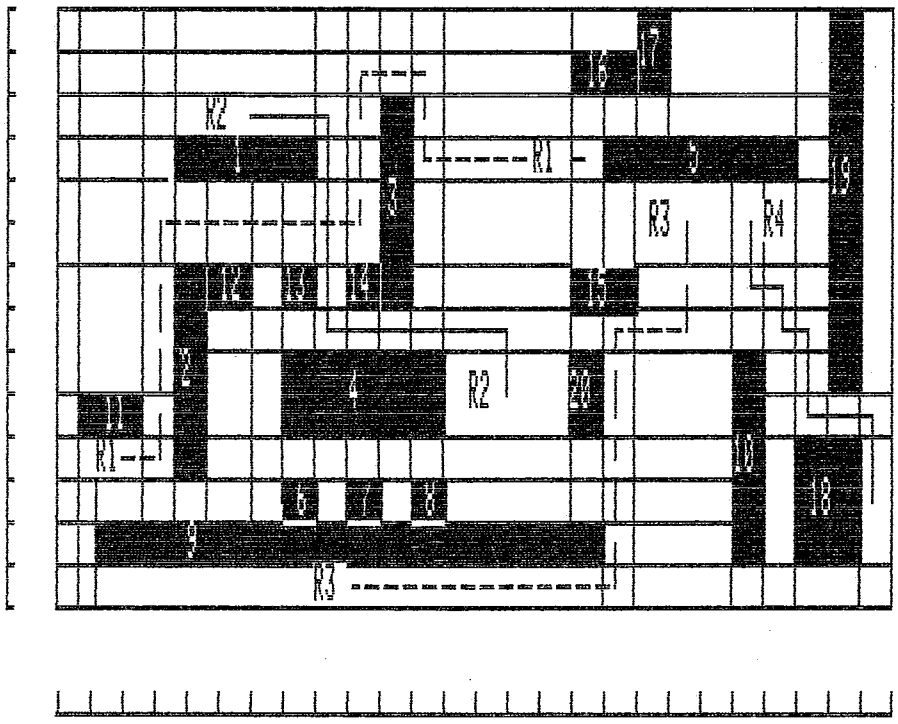


FIGURA VII.7

fatos adicionados : "R1 é uma rota".
"R1 é sensivel ao calor".
"O obstáculo 04 é quente".
"O obstáculo 010 é frio".
"O espaçamento minimo assumido de R1 para qualquer obstáculo é igual a 1".
"O espaçamento mínimo de R1 para 04 deve ser igual a 10".
"A direção Oeste é uma direção proibida para R1".
"E proibida a passagem da rota nas vizinhanças do obstáculo 020".

regras instanciadas : P3, P4.

ponto inicial : (20;35)
ponto objetivo : (165;105)
número de celas da rota : 20
número de celas expandidas : 48
número de celas atingidas : 62
número de celas da configuração : 142
distância manhattam percorrida : 257,5
tempo de geração da rota : 1:32 minutos.

- Rota 2 (R2).

função de avaliação : F = DM + EDM,
fatos adicionados : "R2 é uma rota"
"O espaçamento minimo assumido de R2 para qualquer obstáculo é igual a 1".
"A coincidência parcial ou total de

R2 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (50;115)

ponto objetivo : (140;45)

número de celas da rota : 11

número de celas expandidas : 11

número de celas atingidas : 18

número de celas da configuração : 142

distância manhattam percorrida : 147,5

tempo de geração da rota : 21 segundos.

- Rota 3 (R3).

função de avaliação : F = DM + EDM

fatos adicionados : "R3 é uma rota".

"R3 é sensivel a baixas temperaturas"

"O espaçamento minimo assumido de R3 para qualquer obstáculo é igual a 1".

"A coincidência parcial ou total de R3 com outra rota da configuração é proibida".

"A direção Leste é uma direção proibida para R3".

"E proibida a passagem de R3 nas vizinhanças do obstáculo 020".

regras instanciadas : P4, P3.

ponto inicial : (195;90)

ponto objetivo : (80;5)

nûmero de celas da rota : 10

nûmero de celas expandidas : 30

nûmero de celas atingidas : 35

nûmero de celas da configuraçâo : 142

distância manhattam percorrida : 190

tempo de geraçâo da rota : 1:02 minutos.

- Rota 4 (R4).

função de avaliaçâo : F = DM + EDM

fatos adicionados : "R4 é uma rota".

"R4 é sensivel a baixas temperaturas".

"O espaçamento minimo assumido de R4 para qualquer obstâculo é igual a 1".

"A coincidência parcial ou total de R4 com outra rota da configuraçâo é proibida".

"A direçâo Oeste é uma direçâo proibida para R4".

regras instanciadas : P4, P3.

ponto inicial : (215;90)

ponto objetivo : (255;30)

nûmero de celas da rota : 10

nûmero de celas expandidas : 10

nûmero de celas atingidas : 13

nûmero de celas da configuraçâo : 142

distância manhattam percorrida : 105

tempo de geraçâo da rota : 15 segundos.

7.1.6) Teste 6.

Os resultados deste teste podem ser vistos gràficamente na figura VII.8.

- Rota 1 (R1).

função de avaliação : F = DM + EDM,

DM -> distância manhattam percorrida.

EDM-> estimativa da distância manhattam a ser percorrida desde a cela avaliada até a cela objetivo.

fatos adicionados : "R1 é uma rota".
"R1 é sensivel ao calor".
"O obstáculo 04 é quente".
"O obstáculo 010 é frio".
"O espaçamento minimo assumido de R1 para qualquer obstáculo é igual a 1".
"O espaçamento mínimo de R1 para 04 deve ser igual a 10".
"E proibida a passagem da rota nas vizinhanças do obstáculo 020".

regras instanciadas : P3, P4.

ponto inicial : (20;35)

ponto objetivo : (165;105)

número de celas da rota : 20

número de celas expandidas : 48

número de celas atingidas : 62

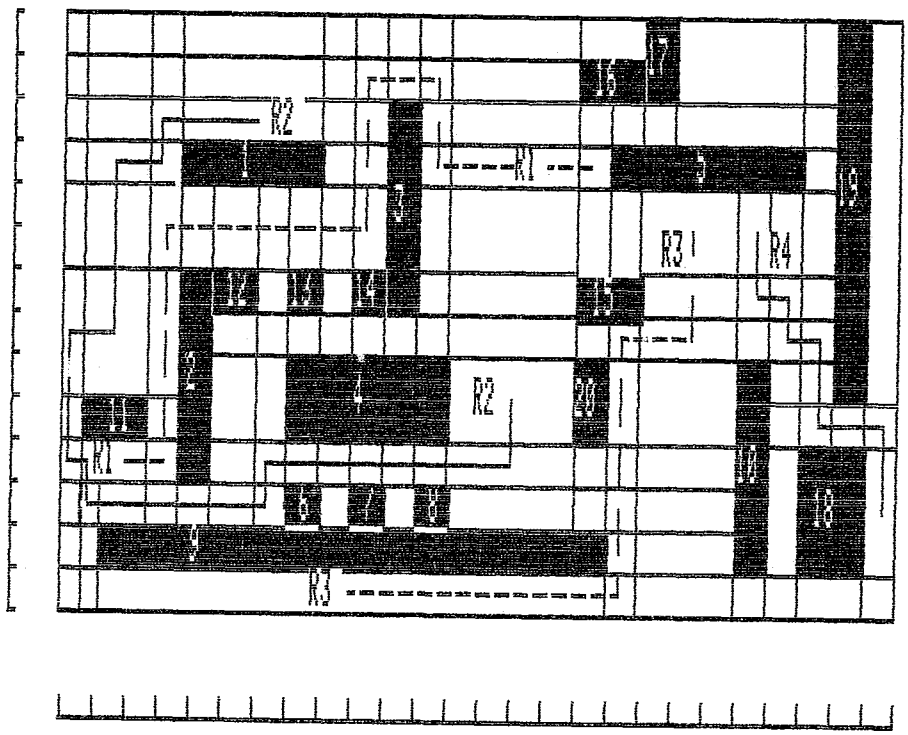


FIGURA VII.8

número de celas da configuração : 142

distância manhattam percorrida : 257,5

tempo de geração da rota : 35 segundos.

- Rota 2 (R2).

função de avaliação : F = DM + EDM,

fatos adicionados : "R2 é uma rota".

"O espaçamento mínimo assumido de R2 para qualquer obstáculo é igual a 1".

"É proibida a passagem da rota nas vizinhanças do obstáculo 019".

"É proibido o cruzamento de R2 com qualquer outra rota da configuração".

"A coincidência parcial ou total de R2 com outra rota da configuração é proibida".

regras instanciadas : P4.

ponto inicial : (50;115)

ponto objetivo : (140;45)

número de celas da rota : 24

número de celas expandidas : 50

número de celas atingidas : 59

número de celas da configuração : 142

distância manhattam percorrida : 307,5

tempo de geração da rota : 1:58 minutos.

- Rota 3 (R3).

função de avaliação : F = DM + EDM

fatos adicionados : "R3 é uma rota".
"R3 é sensivel a baixas temperaturas"
"O espaçamento minimo assumido de R3 para qualquer obstáculo é igual a 1".
"E proibido o cruzamento de R3 com qualquer outra rota da configuração".
"A coincidência parcial ou total de R3 com outra rota da configuração é proibida".
"A direção Leste é uma direção proibida para R3".
"E proibida a passagem de R3 nas vizinhanças do obstáculo 010".

regras instanciadas : P4, P8.

ponto inicial : (195;90)
ponto objetivo : (80;5)
número de celas da rota : 10
número de celas expandidas : 24
número de celas atingidas : 25
número de celas da configuração : 142
distância manhattam percorrida : 190
tempo de geração da rota : 1:15 minutos.

- Rota 4 (R4).

função de avaliação : F = DM + EDM
fatos adicionados : "R4 é uma rota".
"R4 é sensivel a baixas

temperaturas".

"O espaçamento mínimo assumido de R4 para qualquer obstáculo é igual a 1".

"É proibido o cruzamento de R4 com qualquer outra rota da configuração".

"A coincidência parcial ou total de R4 com outra rota da configuração é proibida".

"É proibida a passagem de R4 nas vizinhanças do obstáculo O10".

"A direção Oeste é uma direção proibida para R4".

regras instanciadas : P4, P8.

ponto inicial	: (215;90)
ponto objetivo	: (255;30)
número de celas da rota	: 10
número de celas expandidas	: 10
número de celas atingidas	: 13
número de celas da configuração	: 142
distância manhattam percorrida	: 105
tempo de geração da rota	: 30 segundos.

7.1.7) Teste 7.

Os resultados deste teste podem ser vistos gráficamente na figura VII.9.

- Rota 1 (R1).

— Area de Preferencia

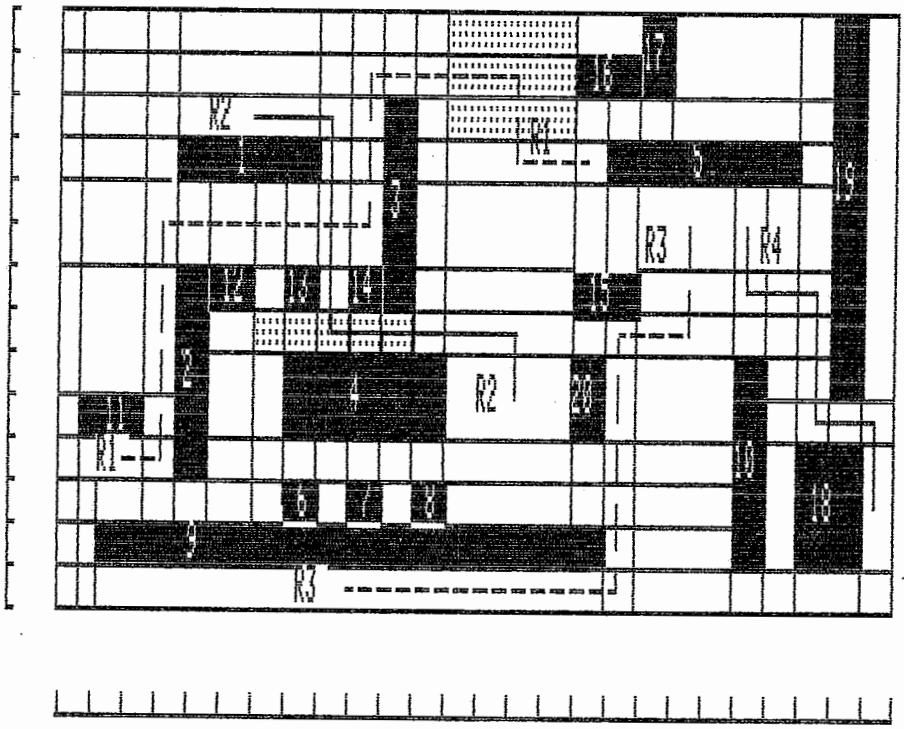


FIGURA VII.9

função de avaliação : F = 0,5(DM + EDM) + 0,5(AP + EAP),

DM -> distância manhattam percorrida.

EDM-> estimativa da distância manhattam a ser percorrida desde a cela avaliada até a cela objetivo.

AP -> medida de performance que serve para atribuir custos totais mais elevados aos segmentos de rota que não se encontram dentro de áreas de preferência.

EAP -> estimativa do custo necessário a se atingir a cela objetivo a partir da cela avaliada, assumindo-se que os segmentos restantes não irão passar por áreas de preferência.

fatos adicionados : "R1 é uma rota".

"R1 é sensivel a baixas temperaturas".

"R1 é favorável à ventilação".

"A1 é uma área".

"A1 é ventilada".

"A1 é uma área de preferência para a

rota R1".

"O obstáculo 04 é quente".

"O obstáculo 010 é pouco frio".

"O espaçamento minimo assumido de R1 para qualquer obstáculo é igual a 1".

"O espaçamento mínimo de R1 para 04 deve ser igual a 10".

"O segmento mínimo de rota de R1 deve ser igual a 15".

"Não é recomendavel a passagem de R1 nas vizinhanças do obstáculo 010".

regras instanciadas : P19, P10, P3, P4.

coordenadas dos centros de gravidade das celas que se encontram em áreas de preferência : (140;135), (140;125), (140;115)

ponto inicial : (20;35)

ponto objetivo : (165;105)

número de celas da rota : 20

número de celas expandidas : 48

número de celas atingidas : 63

número de celas da configuração : 142

distância manhattam percorrida : 257,5

custo devido à função AP : 22,25

custo devido à função DM : 128,75

custo da rota segundo a função de avaliação : 151

tempo de geração da rota : 1:45 minutos.

- Rota 2 (R2).

função de avaliação : F = 0,5(DM + EDM) + 0,5(AP + EAP),

fatos adicionados : "R2 é uma rota"

"R2 é favorável à iluminação".

"O espaçamento mínimo assumido de R2 para qualquer obstáculo é igual a 1".

"A2 é uma área".

"A2 é iluminada".

"A2 é uma área de preferência para R2".

"O segmento mínimo de rota de R2 deve ser igual a 15".

"A coincidência parcial ou total de R2 com outra rota da configuração é proibida".

regras instanciadas : P18, P4.

coordenadas dos centros de gravidade das celas que se encontram em áreas de preferência para R2. : (67;65), (75;65), (85;65), (95;65), (105;65).

ponto inicial : (50;115)

ponto objetivo : (140;45)

número de celas da rota : 11

número de celas expandidas : 11

número de celas atingidas : 19

número de celas da configuração : 142

distância manhattam percorrida : 147,5

custo devido à função AP : 23,5

custo devido à função DM : 64,375

custo da rota segundo a função de: 87,875

avaliação

tempo de geração da rota : 28 segundos.

- Rota 3 (R3).

função de avaliação : F = DM + EDM

fatos adicionados : "R3 é uma rota".

"R3 é sensivel a baixas temperaturas"

"Não é recomendavel a passagem de R3 nas vizinhanças do obstáculo 010".

"O espaçamento minimo assumido de R3 para qualquer obstáculo é igual a 1".

"O segmento mínimo de rota de R3 deve ser igual a 15".

"A coincidência parcial ou total de R3 com outra rota da configuração é proibida".

"A direção Leste é uma direção proibida para R3".

regras instanciadas : P4, P10.

ponto inicial : (195;90)

ponto objetivo : (80;5)

quantidade de interações com o usuário : 1

número de celas da rota : 10

número de celas expandidas : 27

número de celas atingidas : 31

número de celas da configuração : 142

distância manhattam percorrida : 190

tempo de geração da rota : 1:30 minutos.

- Rota 4 (R4).

função de avaliação : F = DM + EDM

fatos adicionados : "R4 é uma rota".

"R4 é sensivel a baixas temperaturas".

"O espaçamento minimo assumido de R4 para qualquer obstáculo é igual a 1".

"O segmento mínimo de rota de R4 deve ser igual a 15".

"Não é recomendavel a passagem de R4 nas vizinhanças do obstáculo 010".

"E proibido o cruzamento de R4 com qualquer outra rota da configuração".

"A coincidência parcial ou total de R4 com outra rota da configuração é proibida".

regras instanciadas : P4, P10.

ponto inicial : (215;90)

ponto objetivo : (255;30)

quantidade de interações com o usuário : 3

número de celas da rota : 10

número de celas expandidas : 10

número de celas atingidas : 13

número de celas da configuração : 142

distância manhattam percorrida : 105

tempo de geração da rota : 1:30 minutos.

7.2 Análise dos Resultados Experimentais e Conclusões.

De uma maneira geral os testes serviram para verificar o comportamento integrado do Sistema já seja nas etapas de geração de celas, redução do conhecimento (a partir de fatos e regras obtidos da base de conhecimento ou de fatos obtidos de forma conversacional com o usuário), geração de frames e sub-frames e utilização das informações contidas nestes no processo de determinação de rotas. A realização de varios testes se justifica pelo fato de se querer observar o desempenho do sistema quando da utilização de funções globais de avaliação variadas em contextos variados.

O Teste 1 serviu para observar o desempenho do processo de busca quando a versão do A* utilizada possue h=0. Neste caso o processo de busca assume fortes características de busca em largura e os resultados obtidos ilustram o fraco desempenho conseguido pois o número de celas atingidas e expandidas é excessivamente grande. No Teste 1 ainda, foi determinada a rota R1 utilizando "via-points" como pontos intermediários. Isto equivale a uma decomposição do problema e os resultados são visivelmente melhores.

No Teste 2 se realizou a determinação de

rotas usando uma função de avaliação com h diferente de nulo. O cômputo da distância manhattam desde a cela avaliada até a cela objetivo é um limite inferior para o valor real de h (h*), dado que devido a presença de obstáculos na configuração, o tamanho real do segmento de rota compreendido entre a cela avaliada e a cela objetivo pode ser superior à computada por h dada a necessidade da rota de se desviar dos obstáculos. Seja então n uma cela qualquer da representação espacial gerada e f(n) = g(n) + h(n) o formato de nossa função de avaliação. E facil perceber que h(n) <= h*(n) para todas as celas da configuração, então nossa versão do A* é admissível. Como além disso podemos perceber que para duas celas vizinhas m e n se cumpre que h(m) - h(n) é menor que custo do arco entre as duas celas, podemos concluir também que nossa versão do A* é consistente (ver capítulo IV) e a função de avaliação utilizada no Teste 2 otimiza o uso da informação disponível. O processo de busca foi, como era de esperar desta vez, mais eficiente que no Teste 1. A rota R1 seguiu porêm desta vez, um percurso diferente do realizado no Teste 1, que inviabilizou o planejamento da rota R2, pois a partir do ponto inicial de R2 qualquer cela vizinha é coincidente com R1, ocassionando uma situação de fracasso para a determinação de R2. Uma maneira de contornar este problema seria a de fazer uma subdivisão de celas abrindo novas possibilidades de passagem para R1 e R2. Esta alternativa não se mostrou atraente devido ao elevado tempo de processamento necessário à geração de celas (23 minutos) em comparação com o tempo de determinação da rota (em nossos testes no máximo 3 minutos). Por esse motivo, uma alternativa simples

seria a de reiniciar o processo de determinação das rotas fazendo do obstáculo O20 uma vizinhança proibida para R1 (ver Testes 5, 6 e 7), deixando desta forma alternativas de passagem para R2. Deixamos desta maneira, a alternativa de gerar uma representação espacial mais precisa para o caso em que se esgotarem todas as possibilidades de tornar sucedida a determinação da rota. Um teste de determinação de R1 com uso da informação de "via-points" também é mostrada no Teste 2 verificando-se um aumento considerável da eficiência do processo de busca com menor número de celas atingidas e expandidas. Para as rotas R3 e R4 o desempenho foi também melhor que no Teste 1.

No Teste 3 foi utilizada uma função de avaliação do tipo f = v*g + w*h, com w >> v. Neste caso f deixa de ser admissível e o processo de busca assume fortes características de busca em profundidade. O tempo de geração da rota R1 foi bem mais rápido que nos testes anteriores com número bem inferior de celas atingidas e expandidas. A solução obtida foi porém ligeiramente mais "cara" que a melhor solução obtida até o momento para R1. O desempenho na determinação da rotas R2 apresentou as mesmas características que o apresentado na determinação de R1. Na determinação da rota R4 o desempenho "empatou" o obtido no Teste 2 quando foi usada uma versão do A* admissível e consistente. Porém, na determinação da rota R3 o resultado piorou significativamente em relação a todos os testes anteriores. Sabemos que para w >> v, não é mais garantida a otimalidade da solução e não nos é possível afirmar que os bons resultados obtidos na determinação de R1, R2 e R4 serão uma constante em outras configurações problema. Porém

podemos dizer que existe uma boa possibilidade de obter "boas" soluções em tempos de processamento econômicos uma vez que a busca com w >> v tende a dar um peso maior ao valor calculado pela função de estimativa de distância manhattam, direcionando a busca na direção da cela objetivo. Existe, não obstante, o risco de que fracassos da busca numa determinada direção, redirecione a busca continuamente tornando a busca exaustiva, piorando o desempenho geral como foi visto na determinação de R3.

No Teste 4 foi utilizada uma função de avaliação do tipo f = v*g + w*h, com w > v. Esta ponderação dos componentes g e h é mais moderada que usada no Teste 3. Se procurou desta forma melhorar a eficiência do processo de busca em relação ao Teste 2, amenizando os problemas possíveis de surgirem quando w >> v. Para direcionar ainda mais a busca usamos a restrição de "direção proibida". Assim por exemplo, se o ponto objetivo ficava a direita do ponto inicial, a direção Oeste era uma direção proibida para a rota. Ficamos conscientes de estar aumentando com esta abordagem, as possibilidades de fracasso na determinação da rota. Neste caso, de uma forma geral tanto o tempo de processamento quanto o número de celas atingidas e expandidas melhorou em relação ao Teste 3. O uso da restrição de "direção proibida", orientando a busca, teve influência na melhora da eficiência da busca.

Os Testes 5, 6 e 7 serviram para ilustrar os casos em que são usados restrições e heurísticas variadas. O Teste 7 ilustra também alguns casos de interação com o projetista durante a determinação da rota, visando dar maior flexibilidade ao processo decisório inerente à determinação

da viabilidade de uma cela. Pôde-se observar que o número de interações foi pequeno em relação ao número de celas atingidas. As rotas obtidas nestes últimos testes mostraram atender a todas as condições do problema. Os resultados obtidos são por si só, auto-explicativos.

Uma observação de caráter geral foi de que as rotas determinadas anteriormente tendem de uma forma geral a limitar o espaço de busca de um caminho, evitando a seleção de celas como segmentos de rota válidos quando estes já fazem parte de outra rota. Em alguns casos também pôde-se observar tempos de geração diferentes em testes diferentes para uma determinada rota apesar do "custo" da solução e o número de celas atingidas e expandidas serem idênticos. Esta diferença se deve à quantidade variável de restrições que foi preciso avaliar para a rota em cada teste e/ou à complexidade envolvida na avaliação do tipo de restrições usadas em cada caso. Um exemplo desta situação são os resultados obtidos na determinação da rota R4 nos Testes 6 e 7.

Para avaliar o Sistema de planejamento de rotas, devemos considerar que o objetivo principal do sistema é possibilitar a obtenção de soluções reais. Assim, considerações sobre eficiência estão subordinadas à condição de se obterem rotas que satisfaçam a todas as condições do problema estabelecidas pelo projetista. Os testes realizados mostraram a capacidade do Sistema de obter soluções que atendam integralmente à definição do problema. O recurso de interação com o projetista foi usado para dar maior flexibilidade ao processo decisório nos casos previstos quando da especificação da base de conhecimento inicial. O

fracasso na determinação de rotas foi contornado, fazendo-se novas tentativas utilizando o conhecimento sobre os motivos do fracasso para restringir o contexto do problema a situações com possibilidades de sucesso. Outra alternativa possível, embora mais dispendiosa computacionalmente como já foi mencionado, é o aumento gradativo da precisão da representação baseada em celas o que é possível de ser realizado através das opções do Sistema.

No transcorrer desta tese concluimos que os métodos que se mostram mais apropiados à incorporação e utilização do conhecimento no planejamento de rotas entre obstáculos são aqueles que utilizam uma representação da configuração espacial baseada em celas. No capitulo VI abordamos superficialmente a extensão do método de planejamento de rotas objeto principal desta tese para outros problemas de determinação de rotas em configurações com obstáculos, utilizando como base os principais métodos de planejamento de rotas existentes na literatura e que foram descritos no capitulo II. A possibilidade de utilizar o Sistema de planejamento de rotas como núcleo de um sistema especialista foi abordada também no capitulo VI.

De uma forma geral pretendemos ter dado uma pequena contribuição na compreensão da natureza do problema de busca do caminho entre obstáculos em contextos reais para as áreas que nos propusemos abordar no inicio da tese e dos principais aspectos a considerar na solução destes. Foi por nós verificado ao fazer o levantamento dos principais métodos existentes na literatura que os métodos que melhor se prezam à solução de problemas reais são aqueles que tem dominio de aplicação bem delimitado. Desta forma concluimos

que ao invés de tentar se chegar a um único método de planejamento de rotas de aplicação geral que utilize técnicas de representação e utilização do conhecimento é mais conveniente tentar adaptar um método desenvolvido para aplicações específicas para o uso de técnicas de I.A.. Ao tomar como base de nosso estudo o método de Yasuhiro descrito no capitulo III, pretendiamos abrir a "caixa preta" que representava inicialmente para nós os mecanismos de inferência, organização e redução do conhecimento, geração e manipulação de estruturas de conhecimento intermediárias ("frames"), integrados no processo de busca do caminho. No capitulo VI foram analisados outros aspectos do planejamento de rotas. Finalmente, na implementação do método de planejamento de rotas tivemos a oportunidade de testar a operacionalidade das ideias expostas.

REFERENCIAS BIBLIOGRAFICAS

ANDERSON, J. R., CORBETT, A.T. e REISER B.J. (1987), Essential Lisp, Addison-Wesley Publishing Company, Inc., pp. 279-297.

BOYSE, J. W. (1979), "Interference Detection Among Solids and Surfaces", Communications of the ACM, vol. 22, No. 1, pp. 3-9.

BROOKS, R.A. (1983), "Solving the Find-Path Problem by Good Representation of Free Space", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-13, No. 3, pp. 190-197.

BROOKS, R.A. e LOZANO-PEREZ, T. (1985), "A Subdivision Algorithm in Configuration Space for Findpath with Rotation", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-15, No. 2, pp. 224-233.

BYUNG, K. K. e SHIN, K. G. (1985), "Minimum-Time Path Planning for Robot Arms and Their Dynamics", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-15, No. 2, pp. 213-223.

CHIEN, R. T., ZHANG, L. e ZHANG, B. (1984), "Planning Collision-Free Paths for Robotic Arm Among Obstacles", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-6, No. 1, pp. 91-96.

FIKES, R. E., HART, P. E. , NILSSON, N. J. (1972), "Some New Directions in Robot Problem Solving", Machine Intelligence, No. 7, pp. 405-430.

FORGY, C. L. (1982), "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence, No. 19, pp. 17-37.

GALAGIN, N. I. e RABINOVICH (1987), "Intelligent Problem Solvers", Cybernetics, Jan. 1987, pp. 279-289.

GOLDEN, B. (1976), "Shortest-Path Algorithms: A Comparison", Operations Research, vol. 24, No. 6, pp. 1164-1168.

HOEL, J. (1976), "Some Variations of Lee's Algorithm", IEEE Transactions on Computers, vol. c-25, No. 1, pp. 19-24.

IGNAT'YEV, M.B., KULAKOV, M. B. e POKROVSDIY, A.M. (1973), "Robot Manipulator control algorithms", Report No. JPRS, NTIS, Springfield, Va.

KUIPERS, B.J. (1975), "A Frame for Frames", Representation and Understanding: Studies in Cognitive Science, Bobrow D. G., Academic Press, pp. 151-184.

LEE, C. Y. (1961), "An Algorithm for Path Connections and Its Aplications", IRE Transactions on Electronic Computers, vol. EC-10, No. 3, pp. 346-365.

LOZANO-PEREZ, T. (1981), "Automatic Planning of Manipulator Transfer Movements", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-11, No. 10, pp. 681-698.

LOZANO-PEREZ, T. (1981), "Robotics", Artificial Intelligence, No. 19, pp. 137-143.

LOZANO-PEREZ, T. (1983), "Spatial Planning: A Configuration Space Approach", IEEE Transactions on Computers, vol. c-32, No. 2, pp. 108-120.

LOZANO-PEREZ, T. e WESLEY M. A. (1979), "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles", Communications of the ACM, vol. 22, No. 10, pp. 560-570.

MICROSOFT CORPORATION & SOFT WAREHOUSE INC. (1986), "Microsoft LISP - Reference Manual", Bellevue, WA - U.S.A.

NEWELL, A. (1969), "Heuristic Programming: Ill-Structured Problems", Progress in Operations Research, Vol. III, ORSA No. 16, Julius & Aronofsky, John Wiley & Sons Inc., New York, pp. 362-414.

NILSSON, J. N., HART P. e BERTRAM R. (1968), "A Formal Basis for the Heuristic Determination of the Cost Paths", IEEE Transactions of Systems, Science and Cybernetics, vol. SSC-4, No. 2, pp. 100-107.

NILSSON, J. N. (1980), Principles of Artificial Intelligence, Tiogo, Palo Alto, California.

PEARL, J. (1983), "Knowledge versus Search: A Quantitative Analysis Using A*", Artificial Intelligence, No. 20, pp. 1-13.

PEARL, J. (1984), "Some Recent Results in Heuristic Search Theory", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-6, No. 1, pp. 1-12.

PINHO, A. A. (1986), "Prova de Teoremas: Refutação", Relatório Técnico ES 98/86, COPPE/UFRJ, Rio de Janeiro, pp. 2-24.

PINHO, A. A. (1986a), "Métodos Informados de Busca", Relatório Técnico ES 93/86, COPPE/UFRJ, Rio de Janeiro.

POHL, I. (1970), "First Results on the effect of error in heuristic search", Machine Intelligence, B. Meltzer e D. Mitchie, Eds., vol. 5, New York, American Elsevier, pp. 219-234.

RAUCH, B. W. e HINDIM (1986), Artificial Intelligence in Business, Science and Industry, vol.1, Fundamentals, Prentice-Hall, pp. 104-128.

RICH, E. (1983), Artificial Intelligence, McGraw Hill, Inc., pp. 25-75, 135-168.

SILVA-TELLES, P. C. (1976), Tubulações Industriais, Livros Técnicos e Científicos Editora S.A., 4a. Edição.

SILVA-TELLES, P. C. (1987), Tubulações Industriais, Materiais, Projeto e Desenho, Livros Técnicos e Científicos Editora S.A., 7a. Edição.

SIMON, H. A. (1983), "Search and Reasoning in Problem Solving", Artificial Intelligence, No. 9, pp. 7-29.

TAYLOR, R. H. (1979), "Planning and Execution of Straight Line Manipulator Trajectories", IBM Journal of Research and Development, vol. 23, No. 4, pp. 424-436.

UDUPA, S. (1977), "Collision Detection and Avoidance in Computer Controlled Manipulators", Tese, California Institute of Technology, Pasadena, California.

ULRICH, R. A. (1987), Robótica, Uma Introdução, Editora Campus Ltda., Rio de Janeiro, pp. 13-26.

YASUHIRO, K., WADA, Y. e KIOUSHI T. (1986), "Knowledge Representation and Utilization for Optimal Route Search", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-16, No. 3, pp. 454-462.

APENDICE A

A.1 LISTAGEM DOS MODULOS DO SISTEMA.

A.1.1 MODULO EXTENDL.LSP.

```
(DEFUN EQLIST (LAMBDA (LST1 LST2)
  ((NULL LST1)
    (NULL LST2) )
  ((NULL LST2) NIL)
  ((NOT (EQ (CAR LST1) (CAR LST2))) NIL)
  ((EQLIST (CDR LST1) (CDR LST2)) T) ))

(DEFUN CONTLST (LST)
 (LENGTH LST) )

(DEFUN TEMPO (DURACAO)
 ((EQ DURACAO 0))
 (TEMPO (SUB1 DURACAO)) )

(DEFUN DOS (LAMBDA (COMMAND)
((NULL COMMAND)
 (EXECUTE "C:COMMAND.COM") )
 (EXECUTE "C:COMMAND.COM" (PACK*   "/C"   COMMAND))(READ-CHAR)
))

(DEFUN S-OU-N-P (LAMBDA (MSG
                   CHAR READ-CHAR RDS WRS)
( ((NULL MSG))
  (FRESH-LINE)
  (WRITE-STRING (PACK* MSG " <S/N> ")) )
(CLEAR-INPUT)
(LOOP
 (SETQ CHAR (STRING-UPCASE (READ-CHAR)))
 ((EQ CHAR 'S)(WRITE-LINE CHAR)T)
 ((EQ CHAR 'N)(WRITE-LINE CHAR)NIL)
 (WRITE-BYTE 7) ) ))

(DEFUN PROMPT-READ-CHAR (LST PROMPT
    READ-CHAR RDS WRS )
  (WRITE-STRING PROMPT)
  (PRINC (LOOP
    ((FIND (STRING-UPCASE (READ-CHAR)) LST)) )) )

(DEFUN WRITE-LINHA (MSG LIN COL)
(SET-CURSOR LIN COL)
(LOOP
((NULL MSG))
(PRIN1 (POP MSG))
(SPACES 1)) )

(DEFUN MEMBERP (ATOM LST)
((NOT(NULL (MEMBER ATOM LST)))T)
(RETURN NIL))

(DEFUN MEMBER-LIST (LST1 LST2)
((NULL LST2)NIL)
```

```
((EQLIST LST1 (CAR LST2))T)
(MEMBER-LIST LST1 (CDR LST2)))

(DEFUN INVERSE ()
(FOREGROUND-COLOR 0)(BACKGROUND-COLOR 7) )

(DEFUN NORMAL ()
(FOREGROUND-COLOR 7)(BACKGROUND-COLOR 0) )

; FUNCOES DE MANIPULACAO DE TELAS

(DEFUN JANELA-D (LIN-I COL-I LIN-F COL-F
                 LIN-W COL-W LST-I LST-C)
((OR (< LIN-F LIN-I)(< COL-F COL-I)))
(SET-CURSOR LIN-I COL-I)
((EQ LIN-I LIN-F)
 (LOOP
  ((> COL-I COL-F))
  (PRINC (ASCII 205))
  (INCQ COL-I) ) )
((EQ COL-I COL-F)
 (LOOP
  ((> LIN-I LIN-F))
  (PRINC (ASCII 186))
  (INCQ LIN-I)) )
(SETQ LST-I (LIST  LIN-I LIN-F))
(LOOP
 ((NULL LST-I))
 (SETQ LIN-W (POP LST-I)
       COL-W (ADD1 COL-I))
 (SET-CURSOR LIN-W COL-W)
 (LOOP
   ((EQ COL-W COL-F))
   (PRINC (ASCII 205))
   (INCQ COL-W) ) )
(SETQ LST-C (LIST COL-I COL-F))
(LOOP
 ((NULL LST-C))
 (SETQ COL-W (POP LST-C)
       LIN-W (ADD1 LIN-I))
 (SET-CURSOR LIN-W COL-W)
 (LOOP
   ((EQ LIN-W LIN-F))
   (SET-CURSOR LIN-W COL-W)
   (PRINC (ASCII 186))
   (INCQ LIN-W) ) )
(SET-CURSOR LIN-I COL-I)
(PRINC (ASCII 201))
(SET-CURSOR LIN-I COL-F)
(PRINC (ASCII 187))
(SET-CURSOR LIN-F COL-I)
(PRINC (ASCII 200))
(SET-CURSOR LIN-F COL-F)
(PRINC (ASCII 188))      )

(DEFUN LIMPA_JANELA (X1 Y1 X2 Y2)
((OR (< X2 X1) (< Y2 Y1)))
(MAKE-WINDOW X1 Y1 (- X2 X1) (- Y2 Y1))
(CLEAR-SCREEN)
```

```
(MAKE-WINDOW 0 0 25 80)    )

(DEFUN LIMPA_TELA ()
 (LIMPA_JANELA 5 1 21 78)
 (SET-CURSOR 3 1)
 (SPACES 78)
 (SET-CURSOR 22 1)
 (SPACES 78) )

(RDS)
```

A.1.2 MODULO FILEIO.LSP.

```
(DEFUN GRAVA_ARQMEM (NOME LST)
  (SETQ ECHO NIL)(WRS (PACK* NOME ".MEM"))
  (LOOP
   ((NULL LST)(PRINT (LIST 'RDS))(WRS)(SETQ ECHO T))
   (PRINT (LIST 'SETQ (CAR LST) (LIST 'QUOTE (EVAL (POP
LST))))) ) )

(DEFUN CRIA-ARQ-LISTAS (NOME LISTAS)
  (SETQ ECHO NIL)(WRS (PACK* NOME ".MEM"))
  (LOOP
   ((NULL LST)(PRINT (LIST 'RDS))(WRS)(SETQ ECHO T))
   (PRINT (POP LISTAS)) ) )

(DEFUN APPEND-TO-EOF (FILE-NAME NOME EXPN)
  (SETQ ECHO NIL)(WRS FILE-NAME T)
  (WRITEPTR (WRITEPTR 'EOF))
  (PRINT (LIST 'SETQ NOME (LIST 'QUOTE EXPN)))
  (WRS)(SETQ ECHO T) )

(DEFUN LE-ARQ (NOME)
 ((RDS NOME)(SETQ ECHO NIL)
  (LOOP
  ((NULL RDS))
  (EVAL (READ)) ) ))

(DEFUN LE-LISTAS (FILE-NAME LISTA
                                    NAME)
(RDS FILE-NAME)
(READPTR 0)
(SETQ LISTA NIL)
(LOOP
  (SETQ NAME (LIST (READ)))
  ((EQ (CAAR NAME) 'RDS)(RDS)LISTA)
  (SETQ LISTA (APPEND LISTA NAME)) ) )

(DEFUN GRAVA-ATOMOS (NOME LST)
(SETQ ECHO NIL)
(WRS (PACK* NOME ".MEM"))
(LOOP
  ((NULL LST)(PRINT '(RDS))(WRS)(SETQ ECHO T))
  (PRINT (LIST 'SETQ (CAR LST) (EVAL (POP LST)))) ) )

(RDS)
```

A.1.3 MODULO GRAPH.LSP.

```
(SETQ COLUMNS 80); Number of columns on your screen.

(SETQ ROWS 24); Number of rows on your screen.

(DEFUN DOT (X-COORD Y-COORD
    LINELENGTH )
  ((AND (< (- X-MAX) X-COORD X-MAX)
(< (- Y-MAX) Y-COORD Y-MAX) )
    (SET-CURSOR (- Y-MAX Y-COORD) (+ X-MAX X-COORD))
    (PRIN1 DOT) ) )

(SETQ X-MAX (TRUNCATE (ADD1 COLUMNS) 2))

(SETQ Y-MAX (TRUNCATE (ADD1 ROWS) 2))

(SETQ DOT '*)

(DEFUN DRAW (NLAMBDA COMMANDS
  (CLEAR-SCREEN)
  (MAPC 'EVAL COMMANDS)
  (SET-CURSOR 0 0) ))

(DEFUN LINE (X1 Y1 X2 Y2
    DELTA-X DELTA-Y SIGN-DELTA-X SIGN-DELTA-Y)
  (SETQ DELTA-X (- X2 X1)
DELTA-Y (- Y2 Y1)
SIGN-DELTA-X (SIGNUM DELTA-X)
SIGN-DELTA-Y (SIGNUM DELTA-Y)
DELTA-X (ABS DELTA-X)
DELTA-Y (ABS DELTA-Y))
  ((< DELTA-Y DELTA-X)
    (SETQ DELTA-Y (+ DELTA-Y DELTA-Y);Gentle slope
  Y2 (- DELTA-Y DELTA-X)
  DELTA-X (- DELTA-X Y2))
    (LOOP
      (DOT X1 Y1)
      ((EQ X1 X2))
      ( ((PLUSP Y2)
  (INCQ Y1 SIGN-DELTA-Y)
  (DECQ Y2 DELTA-X) )
(INCQ Y2 DELTA-Y) )
      (INCQ X1 SIGN-DELTA-X) ) )
  (SETQ DELTA-X (+ DELTA-X DELTA-X);Steep slope
X2 (- DELTA-X DELTA-Y)
DELTA-Y (- DELTA-Y X2))
  (LOOP
    (DOT X1 Y1)
    ((EQ Y1 Y2))
    ( ((PLUSP X2)
(INCQ X1 SIGN-DELTA-X)
(DECQ X2 DELTA-Y) )
      (INCQ X2 DELTA-X) )
    (INCQ Y1 SIGN-DELTA-Y) ) )
```

```
(DEFUN REDUCED-SIN (DEG)
  (/ (* DEG (+ 1324959969 (*  (SETQ  DEG  (*  DEG  DEG))  (+
-67245 DEG))))
     75914915920) )

(DEFUN REDUCED-COS (DEG)
  (SETQ DEG (* DEG DEG))
  (/ (+ 266153374 (* DEG (+ -40518 DEG)))
     266153374) )

(DEFUN SETPOS (X Y)
  (SETQ X-POS X  Y-POS Y) )

(DEFUN TURN (ANGLE)
  (SETQ HEADING (REM (+ HEADING ANGLE) 360)) )

(DEFUN SETHEADING (ANGLE)
  (SETQ HEADING (REM ANGLE 360)) )

(DEFUN PENDOWN ()
  (SETQ PENDOWN T) )

(DEFUN PENUP ()
  (SETQ PENDOWN NIL) )


(DEFUN TURTLE (NLAMBDA COMMANDS
  (SETPOS 0 0)
  (SETHEADING 0)
  (PENDOWN)
  (APPLY 'DRAW COMMANDS) ))

(DEFUN FORWARD (DISTANCE
    X-OLD Y-OLD )
  (SETQ X-OLD X-POS)
  (SETQ Y-OLD Y-POS)
  (INCQ X-POS (ROUND (* DISTANCE (SIN-DEG HEADING))))
  (INCQ Y-POS (ROUND (* DISTANCE (COS-DEG HEADING))))
  ((NOT PENDOWN))
  (LINE X-OLD Y-OLD X-POS Y-POS) )

(DEFUN FORWARD-THEN-TURN (DISTANCE ANGLE)
  (FORWARD DISTANCE)
  (TURN ANGLE) )

(DEFUN IBM-DOT (X-COORD Y-COORD)
  ((AND (< -161 X-COORD 160)
(< -101 Y-COORD 100) )
    (REGISTER 2 (+ 160 X-COORD))
    (REGISTER 3 (- 100 Y-COORD))
    (REGISTER 0 *COLOR*)
    (INTERRUPT 16) ) )

(MOVD 'IBM-DOT 'DOT);Use IBM plot dot routine

(DEFUN SETCOLOR (LAMBDA (COLOR)
  (SETQ *COLOR* (+ 3071 (LENGTH (MEMBER COLOR
'(WHITE RED GREEN BLACK))))) ))
```

```
(SETCOLOR WHITE)

(DEFUN ALPHA-MODE ();Sets up 25 x 80 color alpha mode
  (REGISTER 0 3)
  (INTERRUPT 16)
  (CURSOR-LINES NIL)
  (MAKE-WINDOW 0 0 25 80) )

(DEFUN TURTLE (NLAMBDA COMMANDS
  (IF (NEQ (CADDDR (MAKE-WINDOW)) 40) (GRAPHICS-MODE) )
  (MAKE-WINDOW 0 0 21 40)
  (SETPOS 0 0)
  (SETHEADING 0)
  (PENDOWN)
  (CATCH 'DRIVER (APPLY 'DRAW COMMANDS))
  (MAKE-WINDOW 21 0 4 40)
  (SET-CURSOR 3 0) ))

(DEFUN C-CURVE (DEPTH);"C" curve function
  ((ZEROP DEPTH)
    (FORWARD *LENGTH*) )
  (TURN 45)
  (C-CURVE (SUB1 DEPTH))
  (TURN -90)
  (C-CURVE (SUB1 DEPTH))
  (TURN 45) )

(SETQ *LENGTH* 3)

(DEFUN D-CURVE (DEPTH FLAG);"Dragon" curve function
  ((ZEROP DEPTH)
    (FORWARD *LENGTH*) )
  (IF FLAG (TURN 45) (TURN -45))
  (D-CURVE (SUB1 DEPTH) T)
  (IF FLAG (TURN -90) (TURN 90))
  (D-CURVE (SUB1 DEPTH) NIL)
  (IF FLAG (TURN 45) (TURN -45)) )

(DEFUN GRAPHICS-MODE (LAMBDA NIL
   (REGISTER 0 4)
   (INTERRUPT 16)
   (MAKE-WINDOW 21 0 4 40) ))


(DEFUN ABC (LAMBDA (LST)
        (TRANSX (FIRST LST)) ))

(DEFUN ORDEN (LAMBDA (LST)
        (TRANSY (SECOND LST)) ))

(DEFUN TRANSX (LAMBDA (ATM)
        (+ (+ (* ESCALA ATM) -160) 39) ))

(DEFUN TRANSY (LAMBDA (ATM)
        (+ (+ (* ESCALA ATM) -100) 59) ))
```

```
(DEFUN MOSTRA_OBSTACULOS (LAMBDA (LST
                                  OBSTACULO        O_INICIAL
O_FINAL
                                  A_INICIAL A_FINAL A1 A2 B1
B2 C1 C2 D1 D2)
((NULL LST))
(SETQ OBSTACULO (CAR LST)
      A_INICIAL (ABC (FIFTH OBSTACULO))
      O_INICIAL (ORDEN (FIFTH OBSTACULO))
      A_FINAL   (ABC (THIRD OBSTACULO))
      O_FINAL   (ORDEN (THIRD OBSTACULO))
      A1  (ABC  (SECOND  OBSTACULO))   A2   (ORDEN   (SECOND
OBSTACULO))
      B1  (ABC   (THIRD   OBSTACULO))   B2   (ORDEN   (THIRD
OBSTACULO))
      C1  (ABC  (FOURTH  OBSTACULO))   C2   (ORDEN   (FOURTH
OBSTACULO))
      D1  (ABC   (FIFTH   OBSTACULO))   D2   (ORDEN   (FIFTH
OBSTACULO)) )
(DRAW (LINE  A1 A2 B1 B2)(LINE B1 B2 C1 C2)(LINE  C1  C2  D1
D2)
      (LINE  D1 D2 A1 A2) )
(LOOP
((> O_INICIAL O_FINAL)(MOSTRA_OBSTACULOS (CDR LST)))
(DRAW (LINE A_INICIAL O_INICIAL  A_FINAL O_INICIAL))
(SETQ O_INICIAL (ADD1 O_INICIAL)) ) ))

(DEFUN MOSTRA_CELAS (LAMBDA (LST
                                  C_INICIAL C_FINAL COORD
                                  A_INICIAL A_FINAL A1 A2 B1
B2 C1 C2 D1 D2)
((NULL LST))
(SETQ COORD (SECOND (CAR LST))
      A_INICIAL (ABC (FOURTH COORD))
      O_INICIAL (ORDEN (FOURTH COORD))
      A_FINAL   (ABC (SECOND COORD))
      O_FINAL   (ORDEN (SECOND COORD))
      A1 (ABC (FIRST COORD))  A2 (ORDEN (FIRST COORD))
      B1 (ABC (SECOND COORD)) B2 (ORDEN (SECOND COORD))
      C1 (ABC (THIRD COORD))  C2 (ORDEN (THIRD COORD))
      D1 (ABC (FOURTH COORD)) D2 (ORDEN (FOURTH COORD)) )
(DRAW (LINE  A1 A2 B1 B2)(LINE B1 B2 C1 C2)(LINE  C1  C2  D1
D2)
      (LINE  D1 D2 A1 A2) )
(MOSTRA_CELAS (CDR LST)) ))

(RDS)
```

A.1.4 MODULO GERACELL.LSP.

```
(LOOP (PRIN1 (ASCII 1)) (EVAL (READ)) ((NULL RDS)) )

(DEFUN S-OU-N-P (LAMBDA (MSG
                         CHAR READ-CHAR RDS WRS)
( ((NULL MSG))
  (FRESH-LINE)
  (WRITE-STRING (PACK* MSG " <S/N> ")) )
(CLEAR-INPUT)
(LOOP
 (SETQ CHAR (STRING-UPCASE (READ-CHAR)))
 ((EQ CHAR 'S)(WRITE-LINE CHAR)T)
 ((EQ CHAR 'N)(WRITE-LINE CHAR)NIL)
 (WRITE-BYTE 7) ) ))

(DEFUN LE-ARQ (NOME)
 ((RDS NOME)
  (LOOP
  ((NULL RDS))
  (EVAL (READ)) ) ))


; INICIO DE FUNCOES PARA GERACAO DE CELAS


(DEFUN INCMBR (LAMBDA (OBJ LST "%" LOCAL : "%" N)
  (SETQ N 0)
  (LOOP
    ((NULL (NTH N LST))
      (CONS OBJ LST) )
    ((EQLIST (NTH N LST) OBJ) LST)
    (SETQ N (+ N 1)) ) ))

(DEFUN GERAINT (PRECISAO
                N1 VAR1 VAR2)
  (SETQ OXINTER (LIST (FOURTH LAYOUT) (THIRD LAYOUT)))
  (SETQ OYINTER (LIST (FOURTH LAYOUT) (FIRST LAYOUT) ))
  (SETQ N1 0)
  (LOOP
    ((NULL (NTH N1 OBSTACLE))(SETQ OXINTER  (SORTX  OXINTER)
OYINTER (SORTY
      OYINTER))(IF  (>  PRECISAO  1)  (INTER-POINTS  OXINTER
OYINTER PRECISAO)) )
    (SETQ VAR1 (NTH N1 OBSTACLE))
    (SETQ N1 (+ N1 1))
    (SETQ VAR2 (LIST (CAR (SECOND VAR1)) 0))
    (SETQ OXINTER (INCMBR VAR2 OXINTER))
    (SETQ VAR2 (LIST 0 (SECOND (SECOND VAR1)) ))
    (SETQ OYINTER (INCMBR VAR2 OYINTER))
    (SETQ VAR2 (LIST (CAR (FOURTH VAR1)) 0))
    (SETQ OXINTER (INCMBR VAR2 OXINTER))
    (SETQ VAR2 (LIST 0  (CAR (CDR (FIFTH VAR1))) ))
    (SETQ OYINTER (INCMBR VAR2 OYINTER)) ) )

(DEFUN FIND (LAMBDA (LABL "%" LOCAL : "%" LSTW)
  (SETQ LSTW LSTCELL)
```

```
  (LOOP
    ((NULL LSTW)
      'FAIL )
    ((EQ LABL (CAR (CAR LSTW)))
      (CAR LSTW) )
    (SETQ LSTW (CDR LSTW)) ) ))

(DEFUN INSERT-NUMX (LAMBDA (NUMX LST)
  ((NULL LST)
    (LIST NUMX) )
  ((< (CAR NUMX) (CAR (CAR LST)))
    (CONS NUMX LST) )
  ((NOT (< (CAR NUMX) (CAR (CAR LST))))
    (CONS (CAR LST) (INSERT-NUMX NUMX (CDR LST))) ) ))

(DEFUN SORTX (LAMBDA (LST)
  ((NULL LST) NIL)
  (INSERT-NUMX (CAR LST) (SORTX (CDR LST))) ))

(DEFUN INSERT-NUMY (LAMBDA (NUMY LST)
  ((NULL LST)
    (LIST NUMY) )
  ((< (CAR (CDR NUMY)) (CAR (CDR (CAR LST))))
    (CONS NUMY LST) )
  ((NOT (< (CAR (CDR NUMY)) (CAR (CDR (CAR LST)))))
    (CONS (CAR LST) (INSERT-NUMY NUMY (CDR LST))) ) ))

(DEFUN SORTY (LAMBDA (LST)
  ((NULL LST) NIL)
  (INSERT-NUMY (CAR LST) (SORTY (CDR LST))) ))

(DEFUN OXINT-ATUAL (LAMBDA (LST LABL "%LOCAL" : "%" VARW)
  (SETQ VARW (FIND LABL))
  (SETQ VARW (SECOND (SECOND VARW)))
  (SETQ OXINTER (APPEND OXINTER (LIST VARW))) ))

(DEFUN GERACOOR (LAMBDA (OBJ1 OBJ2 OBJ3 "%LOCAL" :  "%"  VAR
COORD)
  (SETQ VAR 0)
  (SETQ COORD (LIST (LIST (CAR OBJ2)(SECOND OBJ3))
                    (LIST (CAR OBJ1)(SECOND OBJ3))
                     OBJ1 OBJ2))
  ((EQ VAR 0) COORD) ))

(DEFUN NBOUND (LAMBDA (LABL CORDN)
  ((EQ (CAR (CDR (SECOND CORDN))) (CAR (CDR (CAR LAYOUT))))
    '(LT N) )
  ((NOT (EQ  (CAR  (CDR  (SECOND  CORDN)))  (CAR  (CDR  (CAR
LAYOUT)))))
    (+ LABL NUMERO) ) ))

(DEFUN DBOUND (LAMBDA (LABL CORDN)
  ((EQ (CAR (CAR CORDN)) (CAR (CAR LAYOUT)))
    '(LT 0) )
  ((NOT (EQ (CAR (CAR CORDN)) (CAR (CAR LAYOUT))))
    (- LABL 1) ) ))

(DEFUN LBOUND (LAMBDA (LABL CORDN)
```

```
   ((EQ (CAR (SECOND CORDN)) (CAR (SECOND LAYOUT)))
     '(LT L) )
   ((NOT (EQ (CAR (SECOND CORDN)) (CAR (SECOND LAYOUT))))
     (+ LABL 1) ) ))

(DEFUN SBOUND (LAMBDA (LABL CORDN)
   ((EQ (CAR (CDR (THIRD CORDN))) (CAR (CDR (THIRD LAYOUT))))
     '(LT S) )
   ((NOT (EQ (CAR (CDR (THIRD CORDN))) (CAR (CDR (THIRD
LAYOUT)))))
     (- LABL NUMERO) ) ))

(DEFUN GERBOUND (LAMBDA (LABL CORDN "%" LOCAL : "%" VAR
CLBOUND)
   (SETQ NUMERO (- (CONTLST OXINTER) 1))
   (SETQ VAR 0)
   (SETQ NO (NBOUND LABL CORDN))
   (SETQ OE (OBOUND LABL CORDN))
   (SETQ LE (LBOUND LABL CORDN))
   (SETQ SL (SBOUND LABL CORDN))
   (SETQ CLBOUND (CONS NO (CONS LE (CONS SL (CONS OE NIL)))))
   ((EQ VAR 0) CLBOUND) ))

(DEFUN NOR-NEXT (LAMBDA (COORD OBSATM NORTEOBS "%LOCAL" :
"%" NORTEW
     CONDICAO)
   (SETQ CONDICAO 0)
   (SETQ NORTEW (CONS (FIRST OBSATM) (CONS (FIFTH OBSATM)
(CONS (FOURTH
       OBSATM) NIL))))
   ((< (CAR (CDR (FIFTH OBSATM))) (CAR (CDR (CAR COORD))))
NORTEOBS)
   ((OR
       (< (CAR (CAR COORD)) (CAR (FIFTH OBSATM)))
       (> (CAR (SECOND COORD)) (CAR (FOURTH OBSATM))) )
NORTEOBS)
   ((< (SECOND (FIFTH OBSATM)) (SECOND (SECOND NORTEOBS)))
NORTEW)
   ((EQ CONDICAO 0) NORTEOBS) ))

(DEFUN NORTEVIS (LAMBDA (LABL CORDN OBSTCLW "%LOCAL" : "%"
NORTEOBS)
   (SETQ NORTEOBS (CONS '(LT N) (CONS (FIRST LAYOUT) (CONS
(SECOND LAYOUT)
       NIL))))
   (LOOP
     ((NULL OBSTCLW)
       (CAR NORTEOBS) )
     (SETQ NORTEOBS (NOR-NEXT CORDN (CAR OBSTCLW) NORTEOBS))
     (SETQ OBSTCLW (CDR OBSTCLW)) ) ))

(DEFUN LESTE-NEXT (LAMBDA (COORD OBSATM LESTEOBS "%LOCAL" :
"%" LESTEW
     CONDICAO)
   (SETQ CONDICAO 0)
   (SETQ LESTEW (CONS (FIRST OBSATM) (CONS (SECOND OBSATM)
(CONS (FIFTH
       OBSATM) NIL))))
```

```
  ((< (CAR (SECOND OBSATM)) (CAR (SECOND COORD))) LESTEOBS)
  ((OR
      (> (CAR (CDR (SECOND COORD))) (CAR (CDR (SECOND
OBSATM))))
      (< (CAR (CDR (THIRD COORD))) (CAR (CDR (FIFTH
OBSATM)))) )
          LESTEOBS)
  ((< (CAR (SECOND OBSATM)) (CAR (SECOND LESTEOBS))) LESTEW)
  ((EQ CONDICAO 0) LESTEOBS) ))

(DEFUN LESTEVIS (LAMBDA (LABL CORDN OBSTCLW "%LOCAL" : "%"
LESTEOBS)
  (SETQ LESTEOBS (CONS '(LT L) (CONS (SECOND LAYOUT) (CONS
(THIRD LAYOUT)
      NIL))))
  (LOOP
    ((NULL OBSTCLW)
      (CAR LESTEOBS) )
    (SETQ LESTEOBS (LESTE-NEXT CORDN (CAR OBSTCLW)
LESTEOBS))
    (SETQ OBSTCLW (CDR OBSTCLW)) ) ))

(DEFUN OESTE-NEXT (LAMBDA (COORD OBSATM OESTEOBS "%" LOCAL :
"%" OESTEW
    CONDICAO)
  (SETQ CONDICAO 0)
  (SETQ OESTEW (CONS (CAR OBSATM) (CONS (THIRD OBSATM) (CONS
(FOURTH OBSATM)
      NIL))))
  ((> (CAR (THIRD OBSATM)) (CAR (CAR COORD))) OESTEOBS)
  ((OR
      (> (CAR (CDR (CAR COORD))) (CAR (CDR (THIRD OBSATM))))
      (< (CAR (CDR (FOURTH COORD))) (CAR (CDR (FOURTH
OBSATM)))) )
          OESTEOBS)
  ((> (CAR (THIRD OBSATM)) (CAR (SECOND OESTEOBS))) OESTEW)
  ((EQ CONDICAO 0) OESTEOBS) ))

(DEFUN OESTEVIS (LAMBDA (LABL CORDN OBSTCLW "%LOCAL" : "%"
OESTEOBS)
  (SETQ OESTEOBS (CONS '(LT Q) (CONS (FIRST LAYOUT) (CONS
(FOURTH LAYOUT)
      NIL))))
  (LOOP
    ((NULL OBSTCLW)
      (CAR OESTEOBS) )
    (SETQ OESTEOBS (OESTE-NEXT CORDN (CAR OBSTCLW)
OESTEOBS))
    (SETQ OBSTCLW (CDR OBSTCLW)) ) ))

(DEFUN SUL-NEXT (LAMBDA (COORD OBSATM SULOBS "%" LOCAL : "%"
SULW CONDICAO)
  (SETQ CONDICAO 0)
  (SETQ SULW (CONS (FIRST OBSATM) (CONS (SECOND OBSATM)
(CONS (THIRD OBSATM)
      NIL))))
  ((> (CAR (CDR (SECOND OBSATM))) (CAR (CDR (FOURTH
COORD)))) SULOBS)
```

```
  ((OR
      (< (CAR (FOURTH COORD)) (CAR (FIFTH OBSATM)))
      (> (CAR (THIRD COORD)) (CAR (FOURTH OBSATM))) )
SULOBS)
  ((> (CAR (CDR (SECOND OBSATM))) (CAR (CDR (SECOND
SULOBS)))) SULW)
  ((EQ CONDICAO 0) SULOBS) ))

(DEFUN SULVIS (LAMBDA (LABL CORDN OBSTCLW "%LOCAL" : "%"
SULOBS)
  (SETQ SULOBS (CONS '(LT S) (CONS (FOURTH LAYOUT) (CONS
(THIRD LAYOUT) NIL))))
  (LOOP
    ((NULL OBSTCLW)
      (CAR SULOBS) )
    (SETQ SULOBS (SUL-NEXT CORDN (CAR OBSTCLW) SULOBS))
    (SETQ OBSTCLW (CDR OBSTCLW)) ) ))

(DEFUN VISAO (LAMBDA (LABL CORDN "%LOCAL" : "%" VST NV LV OV
SV)
  (SETQ NV (NORTEVIS LABL CORDN OBSTACLE))
  (SETQ OV (OESTEVIS LABL CORDN OBSTACLE))
  (SETQ LV (LESTEVIS LABL CORDN OBSTACLE))
  (SETQ SV (SULVIS LABL CORDN OBSTACLE))
  (SETQ VST (CONS NV (CONS LV (CONS SV (CONS OV NIL)))))
  ((NOT (NULL VST)) VST) ))

(DEFUN GERACELL2 (LAMBDA (OBJ "%LOCAL" : "%" N2 XVAR VCOOR
OBSVIEW CLBOUND
    CELL OXINTERW)
  (SETQ N2 0)
  (SETQ OXINTERW NIL)
  (LOOP
    (SETQ N2 (+ N2 1))
    ((NULL (NTH N2 OXINTER))
      (SETQ OXINTER OXINTERW) )
    (SETQ XVAR (NTH N2 OXINTER))
    (SETQ VCOOR (GERACOOR XVAR (NTH (- N2 1) OXINTER) OBJ))
    (SETQ OXINTERW (APPEND OXINTERW (LIST (CAR VCOOR))))
    (SETQ LBL (+ LBL 1))
    (SETQ CLBOUND (GERBOUND LBL VCOOR))
    (SETQ OBSVIEW (VISAO LBL VCOOR))
    (SETQ CELL (CONS LBL (CONS VCOOR (CONS CLBOUND (CONS
OBSVIEW NIL)))))
    (SETQ LSTCELL (CONS CELL LSTCELL)) ) ))

(DEFUN GERACELL1 (LAMBDA (LST1 LST2 "%LOCAL" : "%" N1 VAR1
VCOOR)
  (SETQ N1 0)
  (SETQ LBL 0)
  (SETQ LSTCELL NIL)
  (LOOP
    (SETQ N1 (+ N1 1))
    ((NULL (NTH N1 LST2))
      (SETQ OXINTER NIL)
      (SETQ OYINTER NIL) )
    (SETQ VAR1 (NTH N1 LST2))
    (GERACELL2 VAR1)
```

```
    (SETQ LST1 (OXINT-ATUAL OXINTER LBL)) ) ))

(DEFUN INTER-POINTS (LST1 LST2 PRECISAO)
(INTER-POINTS-X (CAR (SECOND LST1)) LST1 PRECISAO)
(INTER-POINTS-Y (SECOND (SECOND LST2)) LST2 PRECISAO) )

(DEFUN INTER-POINTS-X (ATM LST1 PRECISAO
                       FATOR ATMW COORD-W)
((NULL ATM))
(SETQ ATMW (CAAR LST1)
      FATOR (FLOOR (/ (- ATM ATMW) PRECISAO))
      COORD-W ATMW)
(LOOP
(SETQ COORD-W (+ FATOR COORD-W))
((>= COORD-W ATM)(INTER-POINTS-X (CAR(THIRD LST1)) (CDR
LST1) PRECISAO) )
(SETQ OXINTER (INSERT-NUMX (LIST COORD-W 0) OXINTER)) ) )

(DEFUN INTER-POINTS-Y (ATM LST2 PRECISAO
                       FATOR ATMW COORD-W)
((NULL ATM))
(SETQ ATMW (SECOND(CAR LST2))
      FATOR (FLOOR (/ (- ATM ATMW) PRECISAO))
      COORD-W ATMW)
(LOOP
(SETQ COORD-W (+ FATOR COORD-W))
((>= COORD-W ATM)(INTER-POINTS-Y (SECOND(THIRD LST2)) (CDR
LST2) PRECISAO) )
(SETQ OYINTER (INSERT-NUMY (LIST 0 COORD-W) OYINTER)) ) )

(DEFUN REMOVE (LAMBDA (OBJ LSTCELW)
  ((NULL LSTCELW) NIL)
  ((EQ (CAR OBJ) (CAR (CAR LSTCELW)))
    (CDR LSTCELW) )
  (CONS (CAR LSTCELW) (REMOVE OBJ (CDR LSTCELW))) ))

(DEFUN CONCATN1 (LAMBDA (OBJ1 "%LOCAL" : "%" LSTW)
  (SETQ LSTW LSTCELL)
  (SETQ LSTCELL NIL)
  (LOOP
    ((NULL LSTW) NIL)
    ((EQ (CAR OBJ1) (CAR (CAR LSTW)))
      (SETQ LSTCELL (APPEND (APPEND LSTCELL (LIST OBJ1))
(CDR LSTW))) )
    (SETQ LSTCELL (APPEND LSTCELL (LIST (CAR LSTW))))
    (SETQ LSTW (CDR LSTW)) ) ))

(DEFUN TRATVZ2 (LAMBDA (OBJ1 LSTV "%" LOCAL : "%" OBJW)
  (SETQ OBJW (CONS (CAR OBJ1) (CONS (SECOND OBJ1) (CONS LSTV
(CONS (FOURTH
      OBJ1) NIL)))))
  (CONCATN1 OBJW) ))

(DEFUN FSUL (LAMBDA (OBJ1 OBJ2LST "%LOCAL" : "%" CELW VIZW)
(LOOP
  ((NULL OBJ2LST))
  (SETQ CELW (POP OBJ2LST))
  (SETQ VIZW (APPEND (LIST OBJ1) (CDR (THIRD CELW))))
```

```
  (TRATVZ2 CELW VIZW) ) ))

(DEFUN FLESTE (LAMBDA (OBJ1 OBJ2LST "%" LOCAL : "%" CELW
VIZW)
(LOOP
  ((NULL OBJ2LST))
  (SETQ CELW (POP OBJ2LST))
  (SETQ VIZW (THIRD CELW))
  (SETQ VIZW (CONS (CAR VIZW) (CONS (SECOND VIZW) (CONS
(THIRD VIZW) (CONS
      OBJ1 NIL)))))
  (TRATVZ2 CELW VIZW) ) ))

(DEFUN FNORTE (LAMBDA (OBJ1 OBJ2LST "%" LOCAL : "%" CELW
VIZW)
(LOOP
  ((NULL OBJ2LST))
  (SETQ CELW (POP OBJ2LST))
  (SETQ VIZW (THIRD CELW))
  (SETQ VIZW (CONS (CAR VIZW) (CONS (SECOND VIZW) (CONS OBJ1
(CONS (NTH
      3 VIZW) NIL)))))
  (TRATVZ2 CELW VIZW) ) ))

(DEFUN FOESTE (LAMBDA (OBJ1 OBJ2LST "%LOCAL" : "%" CELW
VIZW)
(LOOP
  ((NULL OBJ2LST))
  (SETQ CELW (POP OBJ2LST))
  (SETQ VIZW (THIRD CELW))
  (SETQ VIZW (CONS (CAR VIZW) (CONS OBJ1 (CONS (THIRD VIZW)
(CONS (NTH
      3 VIZW) NIL)))))
  (TRATVZ2 CELW VIZW) ) ))

(DEFUN STRATVZ1 (OBJ1 ATM2)
  (FLESTE OBJ1 (ACHA-LESTE-CELAS ATM2))
  (FOESTE OBJ1 (ACHA-OESTE-CELAS ATM2))
  (FSUL OBJ1   (ACHA-SUL-CELAS ATM2)) )

(DEFUN OTRATVZ1 (OBJ1 ATM2)
  (FNORTE OBJ1 (ACHA-NORTE-CELAS ATM2))
  (FSUL OBJ1   (ACHA-SUL-CELAS ATM2))
  (FOESTE OBJ1 (ACHA-OESTE-CELAS ATM2)) )

(DEFUN NTRATVZ1 (OBJ1 ATM2)
  (FLESTE OBJ1 (ACHA-LESTE-CELAS ATM2))
  (FOESTE OBJ1 (ACHA-OESTE-CELAS ATM2))
  (FNORTE OBJ1 (ACHA-NORTE-CELAS ATM2)) )

(DEFUN LTRATVZ1 (OBJ1 ATM2)
  (FNORTE OBJ1 (ACHA-NORTE-CELAS ATM2))
  (FSUL OBJ1   (ACHA-SUL-CELAS ATM2))
  (FLESTE OBJ1 (ACHA-LESTE-CELAS ATM2)) )

(DEFUN ACHA-OESTE-CELAS (ATM2
                         LSTW LISTA)
(SETQ LSTW LSTCELL)
```

```
(LOOP
((NULL LSTW)LISTA)
(IF (EQ ATM2 (SECOND(THIRD(CAR LSTW))))
    (PUSH (POP LSTW) LISTA)(POP LSTW)) ) )

(DEFUN ACHA-LESTE-CELAS (ATM2
                                  LSTW LISTA)
(SETQ LSTW LSTCELL)
(LOOP
((NULL LSTW)LISTA)
(IF (EQ ATM2 (FOURTH(THIRD(CAR LSTW))))
    (PUSH (POP LSTW) LISTA)(POP LSTW)) ) )

(DEFUN ACHA-NORTE-CELAS (ATM2
                                  LSTW LISTA)
(SETQ LSTW LSTCELL)
(LOOP
((NULL LSTW)LISTA)
(IF (EQ ATM2 (THIRD(THIRD(CAR LSTW))))
    (PUSH (POP LSTW) LISTA)(POP LSTW)) ) )

(DEFUN ACHA-SUL-CELAS (ATM2
                                LSTW LISTA)
(SETQ LSTW LSTCELL)
(LOOP
((NULL LSTW)LISTA)
(IF (EQ ATM2 (CAR(THIRD(CAR LSTW))))
    (PUSH (POP LSTW) LISTA)(POP LSTW)) ) )

(DEFUN EQVIS (LAMBDA (OBJ1 OBJ2)
  ((EQLISTV (FOURTH OBJ1) (FOURTH OBJ2)) T) ))

(DEFUN SRECOMB2 (LAMBDA (OBJ1 OBJ2 "%LOCAL" : "%" OBJ1W VIEW
VIZ COOR)
  ((OR (NEQ (CAADDR (SECOND OBJ2))(CAADDR(SECOND OBJ1)))
       (NEQ   (CAR(FOURTH(SECOND   OBJ2)))(CAR(FOURTH(SECOND
OBJ1))) )))
  (SETQ VIEW (FOURTH OBJ1))
  (SETQ CHAVE 1)
  (SETQ VIZ (CONS (CAR (THIRD OBJ1))  (CONS  (SECOND  (THIRD
OBJ1)) (CONS (THIRD
      (THIRD OBJ2)) (CONS (FOURTH (THIRD OBJ1)) NIL)))))
  (SETQ COOR (CONS (CAR (SECOND OBJ1)) (CONS (SECOND (SECOND
OBJ1)) (CONS
      (THIRD (SECOND OBJ2))  (CONS  (FOURTH  (SECOND  OBJ2))
NIL))))
  (SETQ OBJ1W (CONS (CAR OBJ1) (CONS COOR  (CONS  VIZ  (CONS
VIEW NIL)))))
  (SETQ LSTCELL (REMOVE OBJ2 LSTCELL))
  (SETQ LABELS (REMOVELBL (CAR OBJ2) LABELS))
  (CONCATN1 OBJ1W)
  (STRATVZ1 (CAR OBJ1) (CAR OBJ2))
  (SETQ CELULA OBJ1W) ))

(DEFUN SRECOMB1 (LAMBDA (CEL "%LOCAL" : "%" SCEL)
  ((NOT (ATOM (THIRD (THIRD CEL)))) T)
  (SETQ SCEL (FIND (THIRD (THIRD CEL))))
  ((EQVIS CEL SCEL)
```

```
    (SRECOMB2 CEL SCEL) ) ))

(DEFUN NRECOMB2 (LAMBDA (OBJ1 OBJ2 "%LOCAL" : "%OBJ1W" VIEW
VIZ COOR)
  ((OR (NEQ (CAADDR (SECOND OBJ2))(CAADDR(SECOND OBJ1)))
       (NEQ   (CAR(FOURTH(SECOND   OBJ2)))(CAR(FOURTH(SECOND
OBJ1))) )))
  (SETQ VIEW (FOURTH OBJ1) CHAVE 1)
  (SETQ VIZ (CONS (CAR (THIRD OBJ2))  (CONS  (SECOND  (THIRD
OBJ1)) (CONS (THIRD
      (THIRD OBJ1)) (CONS (FOURTH (THIRD OBJ1)) NIL)))))
  (SETQ COOR (CONS (CAR (SECOND OBJ2)) (CONS (SECOND (SECOND
OBJ2)) (CONS
      (THIRD (SECOND OBJ1))  (CONS  (FOURTH  (SECOND  OBJ1))
NIL)))))
  (SETQ OBJ1W (CONS (CAR OBJ1) (CONS COOR  (CONS  VIZ  (CONS
VIEW NIL)))))
  (SETQ LSTCELL (REMOVE OBJ2 LSTCELL))
  (SETQ LABELS (REMOVELBL (CAR OBJ2) LABELS))
  (CONCATN1 OBJ1W LSTCELL)
  (NTRATVZ1 (CAR OBJ1) (CAR OBJ2))
  (SETQ CELULA OBJ1W) ))

(DEFUN NRECOMB1 (LAMBDA (CEL "%LOCAL" ":%" NCEL)
  ((NOT (ATOM (CAR (THIRD CEL)))) T)
  (SETQ NCEL (FIND (CAR (THIRD CEL))))
  ((EQVIS CEL NCEL)
    (NRECOMB2 CEL NCEL) ) ))

(DEFUN ORECOMB2 (LAMBDA (OBJ1 OBJ2 "%LOCAL" : "%" OBJ1W VIEW
VIZ COOR)
((OR (NEQ (CADADR(SECOND OBJ2))(CADADR (SECOND OBJ1)))
     (NEQ    (CADADR(CDR(SECOND    OBJ2)))(CADADR(CDR(SECOND
OBJ1))))) ))
  (SETQ VIEW (FOURTH OBJ1))
  (SETQ CHAVE 1)
  (SETQ VIZ (CONS (CAR (THIRD OBJ1))  (CONS  (SECOND  (THIRD
OBJ1)) (CONS (THIRD
      (THIRD OBJ1)) (CONS (FOURTH (THIRD OBJ2)) NIL)))))
  (SETQ COOR (CONS (CAR (SECOND OBJ2)) (CONS (SECOND (SECOND
OBJ1)) (CONS (THIRD
      (SECOND OBJ1)) (CONS (FOURTH (SECOND OBJ2)) NIL)))))
  (SETQ OBJ1W (CONS (CAR OBJ1) (CONS COOR  (CONS  VIZ  (CONS
VIEW NIL)))))
  (SETQ LSTCELL (REMOVE OBJ2 LSTCELL))
  (SETQ LABELS (REMOVELBL (CAR OBJ2) LABELS))
  (CONCATN1 OBJ1W)
  (OTRATVZ1 (CAR OBJ1) (CAR OBJ2))
  (SETQ CELULA OBJ1W) ))

(DEFUN ORECOMB1 (LAMBDA (CEL "%LOCAL" : "%" OCEL)
  ((NOT (ATOM (FOURTH (THIRD CEL)))) T)
  (SETQ OCEL (FIND (FOURTH (THIRD CEL))))
  ((EQVIS CEL OCEL)
    (ORECOMB2 CEL OCEL) ) ))

(DEFUN LRECOMB2 (LAMBDA (OBJ1 OBJ2 "%LOCAL" : "%" OBJ1W VIEW
VIZ COOR)
```

```
((OR (NEQ (CADADR(SECOND OBJ2))(CADADR (SECOND OBJ1)))
     (NEQ    (CADADR(CDR(SECOND    OBJ2)))(CADADR(CDR(SECOND
OBJ1)))) ))
  (SETQ VIEW (FOURTH OBJ1))
  (SETQ CHAVE 1)
  (SETQ VIZ (CONS (CAR (THIRD OBJ1))  (CONS  (SECOND  (THIRD
OBJ2)) (CONS (THIRD
       (THIRD OBJ1)) (CONS (FOURTH (THIRD OBJ1)) NIL)))))
  (SETQ COOR (CONS (CAR (SECOND OBJ1)) (CONS (SECOND (SECOND
OBJ2)) (CONS
       (THIRD (SECOND OBJ2))  (CONS  (FOURTH  (SECOND  OBJ1))
NIL)))))
  (SETQ OBJ1W (CONS (CAR OBJ1) (CONS COOR  (CONS  VIZ  (CONS
VIEW NIL))))
  (SETQ LSTCELL (REMOVE OBJ2 LSTCELL))
  (SETQ LABELS (REMOVELBL (CAR OBJ2) LABELS))
  (CONCATN1 OBJ1W)
  (LTRATVZ1 (CAR OBJ1) (CAR OBJ2))
  (SETQ CELULA OBJ1W) ))

(DEFUN LRECOMB1 (LAMBDA (CEL "%LOCAL" : "%" LCEL)
  ((NOT (ATOM (THIRD (THIRD CEL)))) T)
  (SETQ LCEL (FIND (SECOND (THIRD CEL))))
  ((EQVIS CEL LCEL)
    (LRECOMB2 CEL LCEL) ) ))

(DEFUN RECOMB2 (LAMBDA (
                        N)
  (SETQ N 1)
  (SETQ CHAVE 0)
  (SETQ LABELS (MARCA LSTCELL))
  (LOOP
    ((NULL LABELS) T)
    (SETQ CELULA (FIND (CAR LABELS)))
    (NRECOMB1 CELULA)
    (SRECOMB1 CELULA)
    (LRECOMB1 CELULA)
    (ORECOMB1 CELULA)
    (SETQ LABELS (REMOVELBL (CAR CELULA) LABELS)) ) ))

(DEFUN RECOMB1 (LAMBDA NIL
  (SETQ CHAVE 1)
  (LOOP
    ((EQ CHAVE 0)T)
    (RECOMB2) ) ))

(DEFUN EQLISTV (LAMBDA (LST1 LST2)
  ((NULL LST1)
    (NULL LST2) )
  ((NULL LST2) NIL)
  ((NOT (EQLIST (CAR LST1) (CAR LST2))) NIL)
  ((EQLISTV (CDR LST1) (CDR LST2)) T) ))

(DEFUN GERA_CELAS (PRECISAO)
  (LE-ARQ 'CONFIG.MEM)
  (GERAINT PRECISAO)
  (GERACELL1 OXINTER OYINTER)
  (OBTVRF)
```

```
  ((< PRECISAO 1)(RECOMB1))
   )

(DEFUN INICIA-GERACAO-CELAS (
                                PRECISAO)
(LIMPA_TELA)
(SET-CURSOR 3 2)(SPACES 30)(WRITE-STRING "GERACAO DE CELAS")
(SET-CURSOR 9 2)(WRITE-STRING "GRAU DE PRECISAO DESEJADO  (0
- N) => ")
(SETQ PRECISAO (RATOM))
(SET-CURSOR 22 2)
(WRITE-LINHA '(GERANDO  CELAS  OBSTACULO-ADAPTATIVAS...)  22
19)
(GERA_CELAS PRECISAO)
(GRAVA_ARQMEM 'CONFIG '(LAYOUT OBSTACLE PRED-OBST LSTCELL))
(LOAD MENUP) )

(DEFUN OELIM (LAMBDA (VAR OBSATM "%LOCAL" : "%" LIMT)
  (SETQ LIMT (CONS (CAR (THIRD VAR))  (CONS  (SECOND  (THIRD
VAR)) (CONS (THIRD
       (THIRD VAR)) (CONS (CAR OBSATM) NIL)))))
  (SETQ VAR1 (CONS (CAR VAR) (CONS (SECOND VAR)  (CONS  LIMT
(CONS (FOURTH VAR)
       NIL)))))  ))

(DEFUN SELIM (LAMBDA (VAR OBSATM "%LOCAL" : "%" LIMT)
  (SETQ LIMT (CONS (CAR (THIRD VAR))  (CONS  (SECOND  (THIRD
VAR)) (CONS (CAR
       OBSATM) (CONS (FOURTH (THIRD VAR)) NIL)))))
  (SETQ VAR1 (CONS (CAR VAR) (CONS (SECOND VAR)  (CONS  LIMT
(CONS (FOURTH VAR)
       NIL)))))  ))

(DEFUN LELIM (LAMBDA (VAR OBSATM "%LOCAL" : "%LIMT")
  (SETQ LIMT (CONS (CAR  (THIRD  VAR))  (CONS  (CAR  OBSATM)
(CONS (THIRD (THIRD
       VAR)) (CONS (FOURTH (THIRD VAR)) NIL)))))
  (SETQ VAR1 (CONS (CAR VAR) (CONS (SECOND VAR)  (CONS  LIMT
(CONS (FOURTH VAR)
       NIL)))))  ))

(DEFUN NELIM (LAMBDA (VAR OBSATM "%LOCAL" : "%" LIMT)
  (SETQ LIMT (CONS (CAR OBSATM) (CONS (SECOND  (THIRD  VAR))
(CONS (THIRD
       (THIRD VAR)) (CONS (FOURTH (THIRD VAR)) NIL)))))
  (SETQ VAR1 (CONS (CAR VAR) (CONS (SECOND VAR)  (CONS  LIMT
(CONS (FOURTH VAR)
       NIL)))))  ))

(DEFUN NOVOLIMT2 (LAMBDA (VAR CELL OBSATM)
  ((EQ (CAR (THIRD VAR)) (CAR CELL))
    (NELIM VAR OBSATM) )
  ((EQ (SECOND (THIRD VAR)) (CAR CELL))
    (LELIM VAR OBSATM) )
  ((EQ (THIRD (THIRD VAR)) (CAR CELL))
    (SELIM VAR OBSATM) )
  ((EQ (FOURTH (THIRD VAR)) (CAR CELL))
    (OELIM VAR OBSATM) ) ))
```

```
(DEFUN NOVOLIMT (LAMBDA (CELL OBSATM "%LOCAL" : "%" LIMT
LSTW1)
  (SETQ LSTW1 LSTCELL)
  (SETQ LSTCELL NIL)
  (LOOP
    ((NULL LSTW1))
    (SETQ VAR1 (CAR LSTW1))
    (NOVOLIMT2 VAR1 CELL OBSATM)
    (SETQ LSTCELL (APPEND LSTCELL (LIST VAR1)))
    (SETQ LSTW1 (CDR LSTW1)) ) ))

(DEFUN ELIMINA (LAMBDA (CELL OBSATM "%LOCAL" : "%" LST1
LSTCLW2)
  (SETQ LSTCLW2 LSTCELL)
  (SETQ LST1 NIL)
  (LOOP
    ((EQ (CAR CELL) (CAR (CAR LSTCLW2)))
      (SETQ LSTCELL (APPEND LST1 (CDR LSTCLW2))) )
    (SETQ LST1 (APPEND LST1 (LIST (CAR LSTCLW2))))
    (SETQ LSTCLW2 (CDR LSTCLW2)) ) ))

(DEFUN NCOINCID (LAMBDA (COORD OBSATM)
;   ((NOT (EQ (SECOND (CAR COORD)) (SECOND (SECOND OBSATM))))
T)
  ((AND
      (>= (CAR (CAR COORD)) (CAR (SECOND OBSATM)))
      (<= (CAR (SECOND COORD)) (CAR (THIRD OBSATM))) )
    (SETQ COUNTW (+ COUNTW 1)) ) ))

(DEFUN LCOINCID (LAMBDA (COORD OBSATM)
;   ((NOT (EQ (CAR (SECOND COORD)) (CAR (THIRD OBSATM)))) T)
  ((AND
      (<= (SECOND (SECOND COORD)) (SECOND (THIRD OBSATM)))
      (>= (SECOND (THIRD COORD)) (SECOND (FOURTH OBSATM))) )
    (SETQ COUNTW (+ COUNTW 1)) ) ))

(DEFUN SCOINCID (LAMBDA (COORD OBSATM)
 ;   ((NOT (EQ (SECOND (FOURTH COORD)) (SECOND (FIFTH
OBSATM)))) T)
  ((AND
      (>= (CAR (FOURTH COORD)) (CAR (FIFTH OBSATM)))
      (<= (CAR (THIRD COORD)) (CAR (FOURTH OBSATM))) )
    (SETQ COUNTW (+ COUNTW 1)) ) ))

(DEFUN OCOINCID (LAMBDA (COORD OBSATM)
;   ((NOT (EQ (CAR (CAR COORD)) (CAR (SECOND OBSATM)))) T)
  ((AND
      (<= (SECOND (CAR COORD)) (SECOND (THIRD OBSATM)))
      (>= (SECOND (FOURTH COORD)) (SECOND (FIFTH OBSATM))) )
    (SETQ COUNTW (+ COUNTW 1)) ) ))

(DEFUN TESTEOBST (LAMBDA (CELULA "%LOCAL" : "%" N)
  (SETQ N 0)
  (LOOP
    (SETQ COUNTW 0)
    ((NULL (NTH N OBSTACLE))T)
    (NCOINCID (SECOND CELULA) (NTH N OBSTACLE))
```

```
    (LCOINCID (SECOND CELULA) (NTH N OBSTACLE))
    (SCOINCID (SECOND CELULA) (NTH N OBSTACLE))
    (OCOINCID (SECOND CELULA) (NTH N OBSTACLE))
    ((> COUNTW 2)
      (ELIMINA CELULA (NTH N OBSTACLE))
      (NOVOLIMT CELULA (NTH N OBSTACLE)) )
    (SETQ N (+ N 1))   )))

(DEFUN OBTVRF (
                CELULA LSTCLW)
  (SETQ LSTCLW LSTCELL)
  (LOOP
    ((NULL LSTCLW))
    (SETQ CELULA (CAR LSTCLW))
    (TESTEOBST CELULA)
    (SETQ LSTCLW (CDR LSTCLW)) ) )

(DEFUN REMOVELBL (LAMBDA (OBJ LST)
  ((NULL LST) NIL)
  ((EQ OBJ (CAR LST))
    (CDR LST) )
  (CONS (CAR LST) (REMOVELBL OBJ (CDR LST))) ))

(DEFUN MARCA (LAMBDA (LST "%LOCAL" : "%LSTW1" LSTW2)
  (SETQ LSTW1 NIL)
  (SETQ LSTW2 LST)
  (LOOP
    ((NULL LSTW2) LSTW1)
    (SETQ LSTW1 (APPEND LSTW1 (LIST (CAR (CAR LSTW2)))))
    (SETQ LSTW2 (CDR LSTW2)) ) ))


(PROGN
(IF (S-OU-N-P "O PROGRAMA ESTA OK ?") (SETQ DRIVER
'INICIA-GERACAO-CELAS))
(LE-ARQ 'EXTENDL.LSP)(LE-ARQ 'FILEIO.LSP)
(WRITE-STRING "PARA SALVAR O PROGRAMA E OS DADOS DA SESSAO
CORRENTE
                DIGITE <SAVE GERACELL>")
(TERPRI 2)
)

(RDS)
```

A.1.5 MODULO INFERENC.LSP.

```
(DEFUN MAPCAR (LAMBDA (FUN LST)
  ((NULL LST) NIL)
  (CONS (EVAL (LIST FUN (QUOTE(CAR LST)))) (MAPCAR FUN  (CDR
LST))) ))

(DEFUN MAPCAN (LAMBDA (FUN LST "%LOCAL" ":%PARCIAL")
  ((NULL LST) NIL)
  (SETQ PARCIAL (EVAL(LIST FUN (QUOTE(CAR LST)))))
  ((EQ PARCIAL NIL)
    (MAPCAN FUN (CDR LST)) )
  (CONS PARCIAL (MAPCAN FUN (CDR LST))) ))


; INICIO DO MODULO DE INFERENCIA

(DEFUN CODIFICA_UMA_REGRA (LAMBDA (REGRA "%LOCAL" :  "%NOME"
CONDICAO ACAO)
  (SETQ NOME (CAR REGRA))
  (SETQ CONDICAO (CADR REGRA))
  (SETQ ACAO (NTH 3 REGRA))
  (CODIFICA_CONDICAO CONDICAO NOME)
  (PUT NOME  'ACAO  (CONSTROI_ACAO  (MAPCAR  'CDR  CONDICAO)
ACAO)) ))

(DEFUN CODIFICA_CONDICAO (LAMBDA (CONDICOES REGRA "%LOCAL" :
"%NPROP"
    WCONDICOES APPLST)
  (SETQ NPROP 1)
  (SETQ WCONDICOES CONDICOES)
  (SET (PACK (LIST 'PP NPROP)) (CAR WCONDICOES))
  (ATUALIZAR REGRA (CAAR  (PACK  (LIST  'PP  NPROP)))  (PACK
(LIST 'PP NPROP)))
  (SETQ APPLST (LIST (CDAR (PACK (LIST 'PP NPROP)))))
  (LOOP
    ((NULL (CDR WCONDICOES))
      (PUT REGRA 'CONDICAO (CONSTROI_CONDICAO APPLST)) )
    (POP WCONDICOES)
    (SETQ NPROP (+ NPROP 1))
    (SET (PACK (LIST 'PP NPROP)) (CAR WCONDICOES))
    (ATUALIZAR REGRA (CAAR (PACK (LIST  'PP  NPROP)))  (PACK
(LIST 'PP NPROP)))
    (SETQ APPLST (APPEND APPLST (LIST (CDAR (PACK (LIST  'PP
NPROP))))))  ) ))

(DEFUN ATUALIZAR (LAMBDA (REGRA PREDICADO NOMEPROP)
  (PUT REGRA NOMEPROP PREDICADO)
  (PUT  PREDICADO  NOMEPROP  (CONS  REGRA  (GET  PREDICADO
NOMEPROP)))
  ((NOT (MEMBER NOMEPROP (GET PREDICADO LABEL_PROPS)))
    (PUT PREDICADO LABEL_PROPS (CONS NOMEPROP (GET PREDICADO
LABEL_PROPS))) )
        ))

(DEFUN CONSTROI_CONDICAO (LAMBDA (N_VARS)
  (APPEND '(LAMBDA (NFACTS)) (CONSTROI_TESTE N_VARS)) ))
```

```
(DEFUN CONSTROI_TESTE (LAMBDA (N_VARS "%LOCAL"  ":%NW1"  NW2
CONDLIST LSTW)
  ((NULL (CDR N_VARS)) T)
  (SETQ NW1 0)
  (SETQ CONDLIST NIL)
  ((NULL N_VARS)
    'FATAL_ERROR )
  (LOOP
    ((AND
        (< (CONTLST CONDLIST) 2)
        (NULL (NTH (+ NW1 1) N_VARS)) ) CONDLIST)
    ((NULL (NTH (+ NW1 1) N_VARS))
      (LIST 'AND CONDLIST) )
    (SETQ NW2 (+ NW1 1))
    (LOOP
      ((NULL (NTH NW2 N_VARS)))
      (SETQ LSTW (CODIFICA_PAR (NTH  NW1  N_VARS)  (NTH  NW2
N_VARS) NW1 NW2))
      (COND
        ((NOT (NULL LSTW))
          (SETQ CONDLIST (APPEND CONDLIST LSTW)) ) )
      (SETQ NW2 (+ NW2 1)) )
    (SETQ NW1 (+ NW1 1)) ) ))

(DEFUN  CODIFICA_PAR  (LAMBDA  (VAR1  VAR2  POSVAR1  POSVAR2
"%LOCAL" : "%N1" N2
    LSTCOND PART1 PART2)
  (SETQ N1 0)
  (SETQ LSTCOND NIL)
  (LOOP
    ((NULL (NTH N1 VAR1)) LSTCOND)
    (SETQ N2 0)
    (LOOP
      ((NULL (NTH N2 VAR2)))
      (COND
        ((EQ (NTH N1 VAR1) (NTH N2 VAR2))
          (SETQ PART1 (LIST 'EQ (LIST  'NTH  N1  (LIST  'NTH
POSVAR1 'NFACTS))))
          (SETQ PART2 (APPEND  PART1  (LIST  (LIST  'NTH  N2
(LIST 'NTH POSVAR2
                'NFACTS)))))
          (SETQ LSTCOND (CONS PART2 LSTCOND)) ) )
      (SETQ N2 (+ N2 1)) )
    (SETQ N1 (+ N1 1)) ) ))

(DEFUN CONSTROI_ACAO (LAMBDA (P_CONDICAO P_ACAO)
  (LIST  'LAMBDA  (LIST  'NFACTS)  (BUILD_ACTS  P_CONDICAO
P_ACAO)) ))

(DEFUN BUILD_ACTS (LAMBDA (P_CONDICOES P_ACAO "%LOCAL" : "%"
ACTION_LIST)
  (SETQ ACTION_LIST (LIST 'LIST (CAR P_ACAO)))
  (SETQ ACTION_LIST (APPEND) ACTION_LIST (GERA_CODIGO (CDR P_
ACAO)
      P_CONDICOES))) ))

(DEFUN GERA_CODIGO  (LAMBDA  (P_ACAO  CONDICOES  "%LOCAL"  :
```

```
"%NW1" NW2 ACTLIST
    P_CONDICAO PACTWK)
  (SETQ P_ACTWK P_ACAO)
  (SETQ ACTLIST NIL)
  (LOOP
    ((NULL P_ACTWK)
      (INVERTE ACTLIST) )
    ((NOT (EQ (SUBSTRING (CAR P_ACTWK) 0 0) '=))
      (APPEND (INVERTE ACTLIST) P_ACTWK) )
    (SETQ NW1 0)
    (LOOP
      ((NULL (NTH NW1 CONDICOES)))
      (SETQ P_CONDICAO (NTH NW1 CONDICOES))
      (SETQ NW2 0)
      (LOOP
        ((NULL (NTH NW2 P_CONDICAO)))
        ((EQ (CAR P_ACTWK) (NTH NW2 P_CONDICAO))
          (SETQ ACTLIST (CONS (LIST 'NTH NW2 (LIST 'NTH  NW1
'NFACTS))
                ACTLIST))
          (SETQ NW1 100) )
        (SETQ NW2 (+ NW2 1)) )
      (SETQ NW1 (+ NW1 1)) )
    (POP P_ACTWK) ) ))

(DEFUN CODIFICA_REGRAS (LAMBDA NIL
  (MAPCAR 'CODIFICA_UMA_REGRA REGRAS) ))

(DEFUN   INTERPRETADOR_DE_REGRAS   (LAMBDA   (NIL   "%LOCAL"
":%CONJUNTO_SOMA"
    CONJUNTO_CONFLITO ESCOLHA)
  (SETQ CONJUNTO_SOMA PRE-CONTEXTO)
  (LOOP
    ((AND
        (NULL CONJUNTO_SOMA)
        (NULL CONJUNTO_CONFLITO) )
      (SETQ CONTEXTO (REMOVE_NIL CONTEXTO)) )
    (COND
      ((NULL CONJUNTO_SOMA)
        (SETQ         ESCOLHA         (RESSOLUCAO_DE_CONFLITOS
CONJUNTO_CONFLITO))
        (EXECUTA_ACAO (CAR ESCOLHA))
        (SETQ CONJUNTO_SOMA (CDAR ESCOLHA))
        (SETQ       CONJUNTO_CONFLITO        (REMOVE_REPETIDOS
CONJUNTO_CONFLITO (CAR
            ESCOLHA))) )
      (T (SETQ CONJUNTO_CONFLITO (APPEND CONJUNTO_CONFLITO
          (ENTRA_CLAUSULA (CAR CONJUNTO_SOMA))))
          (SETQ CONJUNTO_SOMA (CDR CONJUNTO_SOMA))) ) ) ))

(DEFUN RESSOLUCAO_DE_CONFLITOS (LAMBDA (CONJUNTO_CONFLITO)
  (CAR CONJUNTO_CONFLITO) ))

(DEFUN EXECUTA_ACAO (LAMBDA (ACAO)
  (PRINT (CONS 'ATIVANDO ACAO))
  (SETQ CONTEXTO (CONS (CADR ACAO) CONTEXTO)) ))

(DEFUN ENTRA_CLAUSULA  (LAMBDA  (CLAUSULA  "%LOCAL:"  "%NW1"
```

```
PRED ARGS)
  (SETQ PRED (CAR CLAUSULA))
  (SETQ NW1 (GET PRED LABEL_PROPS))
  (SETQ ARGS (CDR CLAUSULA))
  (PUT PRED 'BINDINGS (CONS ARGS (GET PRED 'BINDINGS)))
  (MAPEA_CLAUSULAS ARGS PRED NW1) ))

(DEFUN MAPEA_CLAUSULAS (LAMBDA(ARGS PRED LABELPROP
                                    LST-ARGS     RESPB
INTERME FRESP)
(LOOP
  ((NULL LABELPROP)FRESP)
  (SETQ LST-ARGS (GET PRED (CAR LABELPROP)))
  (SETQ INTERME NIL)
  (LOOP
    ((NULL LST-ARGS))
    (SETQ RESPB (ENTER_NTHPROP ARGS (POP LST-ARGS) (CAR
LABELPROP)))
    (COND
      ((NOT(NULL RESPB))(SETQ INTERME (APPEND INTERME
RESPB))) ) )
  (SETQ FRESP (APPEND FRESP INTERME))
  (POP LABELPROP) ) ))

(DEFUN ENTER_NTHPROP (LAMBDA (ARGS PROD LABELPROP "%LOCAL"
":%CL_LIST")
  (MAPCAN '(LAMBDA (X)
        (MATCH_AND_EXECUTE X PROD) ) (CRUZA_PRODUTO
(MULTI_CLAUSULAS ARGS
             PROD 1 LABELPROP))) ))

(DEFUN MULTI_CLAUSULAS (LAMBDA (ARGS PROD POSITION LABELPROP
"%LOCAL" :
    "%LST" PPW)
  (SETQ PPW (PACK (LIST 'PP POSITION)))
  ((NULL (GET PROD PPW)) NIL)
  ((EQ PPW LABELPROP)
    (CONS (LIST ARGS) (MULTI_CLAUSULAS ARGS PROD (+ POSITION
1)
        LABELPROP)) )
  (SETQ LST (GET (GET PROD PPW) BINDINGS))
  (CONS LST (MULTI_CLAUSULAS ARGS PROD (+ POSITION 1)
LABELPROP)) ))

(DEFUN CRUZA_PRODUTO (LAMBDA (LST_CLAUSULAS "%LOCAL" :
"%LISTA" LISTF)
  ((NULL (CDR LST_CLAUSULAS)) LST_CLAUSULAS)
  (SETQ LISTA (CAR LST_CLAUSULAS))
  (SETQ LISTF NIL)
  (LOOP
    ((NULL LISTA) LISTF)
    (SETQ LISTF (APPEND LISTF (CRUZA_AUXILIAR (LIST (LIST
(CAR LISTA))) (CDR
        LST_CLAUSULAS))))
    (POP LISTA) ) ))

(DEFUN CRUZA_AUXILIAR (LAMBDA (LIST1 LIST2 "%LOCAL" :
"%LISTP" LISTQ LISTO
```

```
    LISTX)
  ((NULL LIST2) LIST1)
  (SETQ LISTP NIL)
  (SETQ LISTO LIST1)
  (LOOP
    ((NULL LISTO)
      (CRUZA_AUXILIAR LISTP (CDR LIST2)) )
    (SETQ LISTQ (CAR LIST2))
    (LOOP
      ((NULL LISTQ))
      (SETQ LISTX  (LIST  (APPEND  (CAR  LISTO)  (LIST  (CAR
LISTQ)))))
      (SETQ LISTP (APPEND LISTP LISTX))
      (POP LISTQ) )
    (POP LISTO) ) ))

(DEFUN MATCH_AND_EXECUTE (LAMBDA (ARGS PROD)
  (COND
    ((APPLY (GET PROD 'CONDICAO) (LIST ARGS))
      (CHECK_FOR_DUPS PROD (APPLY  (GET  PROD  'ACAO)  (LIST
ARGS))) )
    (T NIL) ) ))

(DEFUN CHECK_FOR_DUPS (LAMBDA (PROD ACAO)
  (COND
    ((MEMBER_LST ACAO CONTEXTO) NIL)
    (T (LIST (LIST PROD ACAO))) ) ))

(DEFUN  MEMBER_LST  (LAMBDA  (ACAO  CONTEXTO  "%LOCAL"  :
"%CONTLST")
  (SETQ CONTLST CONTEXTO)
  (LOOP
    ((NULL CONTLST) NIL)
    ((EQLIST ACAO (CAR CONTLST)) T)
    (POP CONTLST) ) ))

(DEFUN  REMOVE_REPETIDOS  (LAMBDA  (CONJUNTO_CONFLITO  ACAO
"%LOCAL" : "%TEMP"
    RESULT)
  (SETQ TEMP CONJUNTO_CONFLITO)
  (SETQ RESULT NIL)
  (LOOP
    ((NULL TEMP) RESULT)
    (COND
      ((NOT (EQLIST (NTH 1 (CAR (CAR TEMP))) (NTH 1 ACAO)))
        (PUSH (CAR TEMP) RESULT) ) )
    (SETQ TEMP (CDR TEMP)) ) ))

(DEFUN REMOVE_NIL (LAMBDA (LST CONTXT NW)
  (SETQ CONTXT NIL)
  (SETQ NW 0)
  (LOOP
    ((NULL (NTH NW CONTEXTO)) CONTXT)
    (COND
      ((NOT (MEMBER NIL (NTH NW CONTEXTO)))
        (SETQ  CONTXT  (APPEND  CONTXT  (LIST  (NTH  NW
CONTEXTO)))) ) )
    (SETQ NW (+ NW 1)) ) ))
```

```
(DEFUN INVERTE (LAMBDA (LST "%LOCAL" : "%LSTW1" LSTW2)
  (SETQ LSTW1 LST)
  (SETQ LSTW2 NIL)
  (LOOP
    ((NULL LSTW1) LSTW2)
    (SETQ LSTW2 (CONS (CAR LSTW1) LSTW2))
    (POP LSTW1) ) ))

(DEFUN MOTOR_DE_INFERENCIA ()
(IF (NEQ DEFINE-AREA 'OK) (DATABAS1))
  (INTERPRETADOR_DE_REGRAS) )

(DEFUN DATABAS1 (LAMBDA NIL
  (SETQ CONTEXTO NIL)
  (SETQ PRE-CONTEXTO (APPEND PRE-CONTEXTO (LE-LISTAS 'FATOS.
MEM 'LST1)))
  (SETQ REGRAS (LE-LISTAS 'REGRAS.MEM 'LST2))
  (CODIFICA_REGRAS) ))

(RDS)
```

A.1.6 MODULO MENUP.LSP.

```
(LOOP (PRINC (ASCII 1))(EVAL (READ)) ((NULL RDS)) )

(DEFUN LE-ARQ (NOME)
 ((RDS NOME)(SETQ ECHO NIL)
  (LOOP
  ((NULL RDS))
  (EVAL (READ)) ) ))

(DEFUN S-OU-N-P (LAMBDA (MSG
                  CHAR READ-CHAR RDS WRS)
( ((NULL MSG))
  (FRESH-LINE)
  (WRITE-STRING (PACK* MSG " <S/N> ")) )
(CLEAR-INPUT)
(LOOP
 (SETQ CHAR (STRING-UPCASE (READ-CHAR)))
 ((EQ CHAR 'S)(WRITE-LINE CHAR)T)
 ((EQ CHAR 'N)(WRITE-LINE CHAR)NIL)
 (WRITE-BYTE 7) ) ))

(DEFUN GERA_CAB_MENU (
                        LINHA2 CAB COLUNAW)
(CLEAR-SCREEN)
(JANELA-D 0 0 23 79)
(SET-CURSOR 0 23)
(WRITE-STRING "  C O P P E  / S I S T E M A S  ")
(WRITE-LINHA '(SOLUCAO DE PROBLEMAS REAIS DE PLANEJAMENTO DE
ROTAS) 1 15)
(SET-CURSOR 2 1)
(TLINHA 78 'S)
(SET-CURSOR 4 1)
(TLINHA 78 'S)
(JANELA-D 21 1 21 78) )

(DEFUN ERRO-MSG (LST)
 ((NULL LST))
 (WRITE-LINHA LST 22 2)
 (TEMPO 50)
 (SET-CURSOR 22 2)
 (SPACES 77) )

(DEFUN TLINHA (TIMES TIPO
                   LINHA CONT)
(SETQ  CONT 1)
(COND
((EQ TIPO 'S)(SETQ TIPO 196))
(T (SETQ TIPO 205)) )
(LOOP
((> CONT TIMES))
(PRINC (ASCII TIPO))
(SETQ CONT (ADD1 CONT)) ) )


(DEFUN MENU_PRINCIPAL (
                          LINHA1      LINHA2      CAB      LEITURA
```

```
ROTA-ANTERIOR RATOM LBL-ROTA
                          LST-AREAS ATTRIB-AREAS)
  (LE-ARQ 'DATABASE.MEM) ;A LISTA "PREDICADOS" ECONTRA-SE EM
DATABASE.MEM
  (LE-ARQ 'CONFIG.MEM)
  (LOOP
    (GERA_CAB_MENU)
    (WRITE-LINHA '(MENU PRINCIPAL) 3 31)
    (WRITE-LINHA '(0 - FINALIZACAO) 8 20)
    (WRITE-LINHA '(1 - AVALIACAO INICIAL DO PROBLEMA) 9 20)
    (WRITE-LINHA   '(2   -   ENTRADA   DE   COORDENADAS   DA
CONFIGURACAO) 10 20)
    (WRITE-LINHA '(3 - GERACAO DE CELAS) 11 20)
    (WRITE-LINHA '(4 - VISUALIZACAO GRAFICA DA CONFIGURACAO)
12 20)
    (WRITE-LINHA '(5 - DETERMINACAO DA ROTA) 13 20)
    (WRITE-LINHA '(6 - ATUALIZACAO DA BASE DE  CONHECIMENTO)
14 20)
    (WRITE-LINHA '(OPCAO =>) 16 20)
    (RATOM)
    ((EQ RATOM 0)
      (CLEAR-SCREEN)
      (GRAVA_ARQMEM 'CONFIG '(LAYOUT OBSTACLE PRED-OBST))
      (SETQ DRIVER 'DRIVER)(RETURN T))
 (COND
  ((EQ RATOM 1)
      (AVALIACAO_PROBLEMA) )
    ((EQ RATOM 2)
      (ENTRADA_COORDENADAS) )
    ((EQ RATOM  3)(GRAVA_ARQMEM  'CONFIG  '(LAYOUT  OBSTACLE
PRED-OBST))
        (LOAD GERACELL))
    ((EQ RATOM 4)
      (LE-ARQ 'CONFIG.MEM)(MOSTRA_CONFIGURACAO))
    ((EQ RATOM 5)
      (GRAVA_ARQMEM  'CONFIG  '(LAYOUT  OBSTACLE   PRED-OBST
LSTCELL))
      (MOSTRA_CONFIGURACAO)
      (LOAD ROUTER2) )
    ((EQ RATOM 6)(ATUALIZ_KNOW_BASE))
    ((OR  (>  RATOM  6)  (<   RATOM   0))(ERRO-MSG   '(OPCAO
INVALIDA))) ) ) )

(DEFUN ENTRADA_COORDENADAS ()
  (WRITE-LINHA '(DESEJA UTILIZAR OS  DADOS  DA  CONFIGURACAO
ANTERIOR ? <S/N>)
   22 2)
  (RATOM)
  (COND
  ((OR (EQ RATOM 'S)(EQ RATOM 's))(LE-ARQ 'CONFIG.MEM))
  (T(SETQ LAYOUT NIL OBSTACLE NIL LSTCELL NIL PRED-OBST NIL)
  (ENTRADA_COORDENADAS_LT)) )
  (MENU-ENTRADA-OBSTACULOS) )

(DEFUN ENTRADA_COORDENADAS_LT (
                                  N LINHAW MENSAGEM  CONTLIN  X
Y)
  (LIMPA_TELA)
```

```
   (WRITE-LINHA '(ENTRADA DE DADOS DA CONFIGURACAO) 3 23)
   (WRITE-LINHA '(COORDENADAS DO LAYOUT -----EIXO-X---EIXO-Y)
6 2)
   (SETQ CONTLIN 7)
   (SETQ MENSAGEM '((SUPERIOR ESQUERDA)
         (SUPERIOR DIREITA)
         (INFERIOR DIREITA)
         (INFERIOR ESQUERDA) ))
   (SETQ LAYOUT NIL)
   (LOOP
     ((NULL MENSAGEM))
     (WRITE-LINHA (CAR MENSAGEM) CONTLIN 2)
     (SET-CURSOR CONTLIN 30)
     (SETQ X (RATOM))
     (SET-CURSOR CONTLIN 40)
     (SETQ Y (RATOM))
     (SETQ LAYOUT (APPEND LAYOUT (LIST (LIST X Y))))
     (SETQ CONTLIN (+ CONTLIN 1))
     (POP MENSAGEM) ) )

(DEFUN GERA-ATRIBUTO-OBST (ATM
                           OPCAO N)
(LIMPA_JANELA 5 1 19 78)
(WRITE-LINHA '(SELECAO DE ATRIBUTOS) 6 29)
(MAKE-WINDOW 9 23 11 32)(INVERSE)
(CLEAR-SCREEN)
(SETQ RATOM NIL N 0)
(LOOP
((NOT (NULL RATOM))
 (NORMAL)(CLEAR-SCREEN)(MAKE-WINDOW 0 0 25 80)
(LIST(LIST (NTH (SUB1 OPCAO) PREDICADOS) ATM)))
   (LOOP
     ((NULL (NTH N PREDICADOS)))
     (COND ((AND(NEQ N 0)(EQ (MOD N 9) 0))(SET-CURSOR 10 0)
                    (PROMPT-READ-CHAR '(C) " - CONTINUA")
                   (CLEAR-SCREEN)(SET-CURSOR 0 0)) )
     (PRIN1 (ADD1 N))(SET-CURSOR ROW 3)
     (PRIN1    '=>)(SPACES     1)(WRITE-LINHA     (LIST(NTH     N
PREDICADOS)) ROW 6)
     (TERPRI 1)
     (INCQ N) )
(WRITE-STRING "OPCAO => ")(SETQ OPCAO (RATOM)) ) )

(DEFUN INCLUI-OBSTACULO (
                         LINHAW MENSAGEM CONTLIN LABL_OBST X
Y COORDW)
   (LIMPA_TELA)
   (WRITE-LINHA '(ENTRADA DE DADOS DA CONFIGURACAO) 3 23)
  (WRITE-LINHA '(DIGITE O NUMERO DE
                 SEQUENCIA A SER ATRIBUIDO AO OBSTACULO =>) 7
2)
   (SETQ LABL_OBST (RATOM))
   ((EQ LABL_OBST '/))
   (SETQ CONTLIN 9)
   (LOOP
     ((EQ LABL_OBST '/))
     (WRITE-LINHA       '(COORDENADAS        DO        OBSTACULO
-----EIXO-X---EIXO-Y) 9 2)
```

```
    (SETQ MENSAGEM '((SUPERIOR ESQUERDA)
          (SUPERIOR DIREITA)
          (INFERIOR DIREITA)
          (INFERIOR ESQUERDA) ))
    (SETQ COORDW NIL)
    (INCQ CONTLIN)
    (LOOP
      ((NULL MENSAGEM))
      (WRITE-LINHA (POP MENSAGEM) CONTLIN 2)
      (SET-CURSOR CONTLIN 33)
      (SETQ X (RATOM))
      (SET-CURSOR CONTLIN 43)
      (SETQ Y (RATOM))
      (SETQ CONTLIN (+ CONTLIN 1))
      (SETQ COORDW (APPEND COORDW (LIST (LIST X Y)))) )
    (WRITE-LINHA '(CONFIRMA DADOS DO OBSTACULO ?  <S/N>  =>)
22 2)
    (RATOM)
    (COND
      ((EQ RATOM 'S)
(SETQ OBSTACLE (INSERE-OBSTACULO (CONS (LIST  'O  LABL_OBST)
COORDW) OBSTACLE))
    (SETQ PRED-OBST  (APPEND  PRED-OBST  (GERA-ATRIBUTO-OBST
(PACK*
          'O LABL_OBST))))
    (SETQ PRED-OBST (APPEND) PRED-OBST (LIST (LIST 'OBSTACULO
      (PACK* 'O LABL_OBST)))))  ))
    (LIMPA_TELA)
    (WRITE-LINHA '(ENTRADA DE DADOS DA CONFIGURACAO) 3 23)
    (WRITE-LINHA  '(DIGITE  O  NUMERO  DE  SEQUENCIA  A  SER
ATRIBUIDO AO OBSTACULO
        =>) 7 2)
    (SETQ LABL_OBST (RATOM))
    (SETQ CONTLIN 9) ) )

(DEFUN INSERE-OBSTACULO (LST1 LST2)
((NULL LST1))
((NULL LST2)(LIST LST1))
((> (CADAAR LST2) (CADAR LST1))(CONS LST1 LST2))
(CONS (CAR LST2) (INSERE-OBSTACULO LST1 (CDR LST2))) )

(DEFUN ATUALIZA-OBSTACULO (
                           LINHAW MENSAGEM CONTLIN LABL_OBST
X Y COORDW COORD
                           ATUALZ OBSTACULO COORDB)
  (LIMPA_TELA)
  (WRITE-LINHA '(ATUALIZACAO DE DADOS DA CONFIGURACAO) 3 20)
 (WRITE-LINHA '(DIGITE O NUMERO DE
               SEQUENCIA DO OBSTACULO =>) 7 2)
  (SETQ LABL_OBST (RATOM))
  ((EQ LABL_OBST '/))
  (SETQ CONTLIN 9)
  (LOOP
    ((EQ LABL_OBST '/))
    (SET-CURSOR 9 2)
    (WRITE-STRING "ATUALIZA COORDENADAS <S/N>?")
       (IF (EQ (RATOM) 'S) (SETQ ATUALZ 'S))
    (LIMPA_JANELA 5 1 19 78)
```

```
  (SETQ OBSTACULO (FIND (PACK* 'O LABL_OBST) OBSTACLE
          '(LAMBDA (X Y)(EQL X (PACK(CAR Y))))))
    (WRITE-LINHA      '(COORDENADAS       DO       OBSTACULO
------EIXO-X---EIXO-Y) 9 2)
    (SETQ MENSAGEM '((SUPERIOR ESQUERDA)
          (SUPERIOR DIREITA)
          (INFERIOR DIREITA)
          (INFERIOR ESQUERDA) ))
    (SETQ COORD (CDR OBSTACULO)
          COORDB COORD
          COORDW NIL)
    (INCQ CONTLIN)
    (LOOP
      ((NULL MENSAGEM))
      (WRITE-LINHA (POP MENSAGEM) CONTLIN 2)
      (SET-CURSOR CONTLIN 33)(PRIN1 (CAR (CAR COORD)))
      (IF (EQ ATUALZ 'S)
        (PROGN
         (SET-CURSOR CONTLIN 33)
         (SETQ X (RATOM)) ))
   (SET-CURSOR CONTLIN 43)(PRIN1 (SECOND (POP COORD)))
   (IF (EQ ATUALZ 'S)
     (PROGN
       (SET-CURSOR CONTLIN 43)
       (SETQ Y (RATOM))
       (SETQ COORDW (APPEND COORDW (LIST (LIST X Y)))) ))
      (SETQ CONTLIN (+ CONTLIN 1))  )
    (SET-CURSOR 19 2)
    (WRITE-STRING  "PROPRIEDADE  :   ")(PRINT   (FIND-ATTRIB
(PACK* 'O LABL_OBST)
           PRED-OBST))
    (WRITE-LINHA '(CONFIRMA ATUALIZACAO ? <S/N> =>) 22 2)
    (RATOM)
    (COND
      ((EQ RATOM 'S)
    (SETQ OBSTACLE (REMOVE OBSTACULO OBSTACLE 'EQUAL))
    (SETQ  OBSTACLE   (INSERE-OBSTACULO   (CONS   (LIST   'O
LABL_OBST)
            (IF (EQ ATUALZ 'S) COORDW COORDB) ) OBSTACLE))
    (SETQ PRED-OBST (REMOVE (FIND (PACK* 'O LABL_OBST)
                    PRED-OBST '(LAMBDA (X Y)(EQL  X  (SECOND
Y)))) PRED-OBST
                    'EQUAL))
    (SETQ PRED-OBST (REMOVE (FIND (PACK* 'O LABL_OBST)
                    PRED-OBST '(LAMBDA (X Y)(EQL  X  (SECOND
Y)))) PRED-OBST
                    'EQUAL))
    (SETQ PRED-OBST  (APPEND  PRED-OBST  (GERA-ATRIBUTO-OBST
(PACK*
          'O LABL_OBST))))
    (SETQ PRED-OBST (APPEND PRED-OBST (LIST (LIST 'OBSTACULO
      (PACK* 'O LABL_OBST)))))  ))
    (LIMPA_TELA)
    (WRITE-LINHA '(ATUALIZACAO DE DADOS DA  CONFIGURACAO)  3
20)
    (WRITE-LINHA '(DIGITE O NUMERO DE SEQUENCIA DO OBSTACULO
        =>) 7 2)
    (SETQ LABL_OBST (RATOM))
```

```
    (SETQ CONTLIN 9) ) )

(DEFUN CONSULTA-OBSTACULO (
                                CONTLIN POS OBST)
(LIMPA_TELA)(SET-CURSOR 5 2)
(WRITE-STRING "ROTULO  V.S.E.    V.S.D.    V.I.D.     V.I.E.
TIPO")
(MAKE-WINDOW 7 2 14 76)(INVERSE)(CLEAR-SCREEN)
(SETQ CONTLIN 0 POS 0)
(LOOP
(SETQ OBST (NTH POS OBSTACLE))
((NULL OBST)(NORMAL)
(MAKE-WINDOW 0  0  25  80)(WRITE-LINHA  '(FINALIZA  <F>)  22
63)(RATOM) )
(COND
((EQ CONTLIN 13)(MAKE-WINDOW 0 0 25 80)(NORMAL)
                        (WRITE-LINHA '(CONTINUA ) 22 63)
                  (RATOM)(SETQ   CONTLIN   0)(SET-CURSOR    22
63)(SPACES 15)
                  (MAKE-WINDOW 7 2 14 76)
                  (INVERSE)(CLEAR-SCREEN)) )
(SET-CURSOR CONTLIN 2)(PRIN1 (PACK (CAR OBST)))
(SET-CURSOR CONTLIN 8)(PRIN1 (CAR(SECOND OBST)))(SPACES 1)
                        (PRIN1 (SECOND(SECOND OBST)))
(SET-CURSOR CONTLIN 18)(PRIN1 (CAR (THIRD OBST)))(SPACES 1)
                        (PRIN1 (SECOND (THIRD OBST)))
(SET-CURSOR CONTLIN 28)(PRIN1 (CAR (FOURTH OBST)))(SPACES 1)
                        (PRIN1 (SECOND (FOURTH OBST)))
(SET-CURSOR CONTLIN 38)(PRIN1 (CAR (FIFTH OBST)))(SPACES 1)
                        (PRIN1 (SECOND (FIFTH OBST)))
(SET-CURSOR CONTLIN 48)
(PRIN1 (FIND-ATTRIB (PACK (CAR OBST)) PRED-OBST))
(INCQ CONTLIN)(INCQ POS) ) )

(DEFUN EXCLUE-OBSTACULO (
                          LINHAW MENSAGEM CONTLIN LABL_OBST X
Y COORDW COORD
                          ATUALZ OBSTACULO COORDB)
  (LIMPA_TELA)
  (WRITE-LINHA '(EXCLUSAO DE DADOS DA CONFIGURACAO) 3 20)
 (WRITE-LINHA '(DIGITE O NUMERO DE
                SEQUENCIA DO OBSTACULO =>) 7 2)
  (SETQ LABL_OBST (RATOM))
  ((EQ LABL_OBST '/))
  (SETQ CONTLIN 9)
  (LOOP
    ((EQ LABL_OBST '/))
    (SET-CURSOR 9 2)
  (SETQ OBSTACULO (FIND (PACK* '0 LABL_OBST) OBSTACLE
          '(LAMBDA (X Y)(EQL X (PACK(CAR Y))))))
    (WRITE-LINHA      '(COORDENADAS       DO       OBSTACULO
------EIXO-X---EIXO-Y) 9 2)
    (SETQ MENSAGEM '((SUPERIOR ESQUERDA)
          (SUPERIOR DIREITA)
          (INFERIOR DIREITA)
          (INFERIOR ESQUERDA) ))
    (SETQ COORD (CDR OBSTACULO)
          COORDB COORD
```

```
          COORDW NIL)
     (INCQ CONTLIN)
     (LOOP
       ((NULL MENSAGEM))
       (WRITE-LINHA (POP MENSAGEM) CONTLIN 2)
       (SET-CURSOR CONTLIN 33)(PRIN1 (CAR (CAR COORD)))
       (SET-CURSOR CONTLIN 43)(PRIN1 (SECOND (POP COORD)))
       (SETQ CONTLIN (+ CONTLIN 1))  )
     (SET-CURSOR 19 2)
     (WRITE-STRING  "PROPRIEDADE  :   ")(PRINT   (FIND-ATTRIB
(PACK* '0 LABL_OBST)
           PRED-OBST))
     (WRITE-LINHA '(CONFIRMA EXCLUSAO ? <S/N> =>) 22 2)
     (RATOM)
     (COND
       ((EQ RATOM 'S)
     (SETQ OBSTACLE (REMOVE OBSTACULO OBSTACLE 'EQUAL))
     (SETQ PRED-OBST (REMOVE (FIND (PACK* '0 LABL_OBST)
                     PRED-OBST '(LAMBDA (X Y)(EQL  X  (SECOND
Y)))) PRED-OBST
                     'EQUAL))
     (SETQ PRED-OBST (REMOVE (FIND (PACK* '0 LABL_OBST)
                     PRED-OBST '(LAMBDA (X Y)(EQL  X  (SECOND
Y)))) PRED-OBST
                     'EQUAL)) ))
     (LIMPA_TELA)
     (WRITE-LINHA '(EXCLUSAO DE DADOS DA CONFIGURACAO) 3 20)
     (WRITE-LINHA '(DIGITE O NUMERO DE SEQUENCIA DO OBSTACULO
         =>) 7 2)
     (SETQ LABL_OBST (RATOM))
     (SETQ CONTLIN 9) ) )


(DEFUN FIND-ATTRIB (ATM LST)
(LOOP
((NULL LST))
((AND  (EQ  ATM   (SECOND   (CAR   LST)))(NEQ   (CAAR   LST)
'OBSTACULO)) (CAAR LST))
(POP LST) ) )

(DEFUN MENU-ENTRADA-OBSTACULOS ()
(SETQ RATOM NIL)
(LOOP
;(JANELA-D 8 17 18 65)
(LIMPA_TELA)
(WRITE-LINHA '(ENTRADA DE COORDENADAS DA CONFIGURACAO) 3 23)
(WRITE-LINHA '(0 - FINALIZACAO) 9 21)
(WRITE-LINHA '(1 - INCLUSAO DE OBSTACULOS) 10 21)
(WRITE-LINHA '(2 - EXCLUSAO DE OBSTACULOS) 11 21)
(WRITE-LINHA '(3 - ATUALIZACAO DE DADOS SOBRE OBSTACULOS) 12
21)
(WRITE-LINHA '(4 - CONSULTA DOS DADOS SOBRE  OBSTACULOS)  13
21)
(WRITE-LINHA '(OPCAO :) 15 24)
(RATOM)
((EQ RATOM 0))
(COND
((EQ RATOM 1)(INCLUI-OBSTACULO))
```

```
((EQ RATOM 2)(EXCLUE-OBSTACULO))
((EQ RATOM 3)(ATUALIZA-OBSTACULO))
((EQ RATOM 4)(CONSULTA-OBSTACULO)) )
(LIMPA_JANELA 5 2 20 78)    )   )

(DEFUN MOSTRA_CONFIGURACAO ()
 (GRAPHICS-MODE)
 (GERA_ESCALA)
 (GRAVA-ATOMOS 'ESCALA '(ESCALA))
 (MOSTRA_REGUA)
 (MOSTRA_LAYOUT)
 (MOSTRA_OBSTACULOS OBSTACLE)
 (MOSTRA_CELAS LSTCELL) )

(DEFUN GERA_ESCALA (LAMBDA (NIL
                                FAT1 FAT2 SCAL1 SCAL2)
 (SETQ FAT1 (ABS (- (FIRST (SECOND  LAYOUT))  (FIRST  (FIRST
LAYOUT))))
       FAT2 (ABS (- (SECOND (FIRST LAYOUT)) (SECOND  (FOURTH
LAYOUT))))
       SCAL1 (/ 160 FAT1)
       SCAL2 (/ 160 FAT2))
 (COND
  ((>= SCAL1 SCAL2)(SETQ ESCALA (FLOOR SCAL2)))
 (T(SETQ ESCALA (FLOOR SCAL1))) )
 ((< ESCALA 1)(SETQ ESCALA 1)) ))


(DEFUN MOSTRA_LAYOUT (LAMBDA NIL
  (DRAW (LINE (ABC (FIRST LAYOUT))(ORDEN (FIRST LAYOUT))
              (ABC (SECOND LAYOUT))(ORDEN (SECOND LAYOUT)) )
        (LINE (ABC (SECOND LAYOUT))(ORDEN (SECOND LAYOUT))
              (ABC (THIRD LAYOUT))(ORDEN (THIRD LAYOUT)) )
        (LINE (ABC (THIRD LAYOUT))(ORDEN (THIRD LAYOUT))
              (ABC (FOURTH LAYOUT))(ORDEN (FOURTH LAYOUT)) )
        (LINE (ABC (FOURTH LAYOUT))(ORDEN (FOURTH LAYOUT))
              (ABC (FIRST LAYOUT))(ORDEN (FIRST LAYOUT)) )
         ) ))

(DEFUN MOSTRA_REGUA (LAMBDA (NIL
                      XI_HT YI_HT XF_HT YF_HT
                      XI_VT YI_VT XF_VT YF_VT )
(SETQ XI_HT -121 YI_HT  -56
      XF_HT  159 YF_HT  -56
      XI_VT -141 YI_VT  -41
      XF_VT -141 YF_VT   99)
(DRAW (LINE XI_HT YI_HT XF_HT YF_HT)
      (LINE XI_VT YI_VT XF_VT YF_VT) )
(DRAW (LINE XI_HT YI_HT XI_HT (+ YI_HT 6)))
(GERA_ESCALAX (/ (- XF_HT XI_HT) 10) ESCALA XI_HT YI_HT)
(DRAW (LINE XI_VT YI_VT (+ XI_VT 6) YI_VT))
(GERA_ESCALAY (/ (- YF_VT YI_VT) 10) ESCALA XI_VT YI_VT) ))

(DEFUN GERA_ESCALAX (LAMBDA (NUM SCALE XOORD YOORD
                                 DELTA-X AB1 ORD5 ORD10 ORD1)
((EQ NUM 0))
(SETQ DELTA-X 2
      AB1 XOORD
```

```
      ORD1 (+ YOORD 3)
      ORD10 (+ YOORD 6))
(LOOP
(SETQ AB1 (+ AB1 (* SCALE 2)) )
((EQ DELTA-X 10)(DRAW (LINE AB1 YOORD AB1 ORD10))
                (GERA_ESCALAX (SUB1 NUM) SCALE AB1 YOORD))
(DRAW (LINE AB1 YOORD AB1 ORD1))
(SETQ DELTA-X (+ DELTA-X 2))  ) ))

(DEFUN GERA_ESCALAY (LAMBDA (NUM SCALE XOORD YOORD
                             DELTA-Y ORDA ABS1 ABS5 ABS10)
((EQ NUM 0))
(SETQ DELTA-Y 2
      ORDA YOORD
      ABS1 (+ XOORD 3)
      ABS10 (+ XOORD 6))
(LOOP
(SETQ ORDA (+ ORDA (* SCALE 2)))
((EQ DELTA-Y 10)(DRAW (LINE XOORD ORDA ABS10 ORDA))
                (GERA_ESCALAY (SUB1 NUM) SCALE XOORD ORDA))
(DRAW (LINE XOORD ORDA ABS1 ORDA))
(SETQ DELTA-Y (+ DELTA-Y 2)) ) ))


(PROGN
((S-OU-N-P    "O     PROGRAMA     ESTA     OK     ?")(LE-ARQ
'EXTENDL.LSP)(LE-ARQ 'FILEIO.LSP)
(LE-ARQ 'GRAPH.LSP)
(SETQ DRIVER 'MENU_PRINCIPAL)
(WRITE-STRING  "PARA  SALVAR  O  ESTADO  ATUAL  DO  PROGRAMA
DIGITAR (SAVE MENUP)")
(TERPRI 2)
(RDS)) )

(RETURN)
```

A.1.7 MODULO ROUTER1.LSP.

```
(DEFUN LE-ARQ (NOME)
 ((RDS NOME)(SETQ ECHO NIL)
  (LOOP
  ((NULL RDS))
  (EVAL (READ)) ) ))

(DEFUN S-OU-N-P (LAMBDA(MSG
                          CHAR READ-CHAR RDS WRS)
( ((NULL MSG))
  (FRESH-LINE)
  (WRITE-STRING (PACK* MSG " <S/N> ")) )
(CLEAR-INPUT)
(LOOP
 (SETQ CHAR (STRING-UPCASE (READ-CHAR)))
 ((EQ CHAR 'S)(WRITE-LINE CHAR)T)
 ((EQ CHAR 'N)(WRITE-LINE CHAR)NIL)
 (WRITE-BYTE 7) ) ))


; INICIO DO MODULO DE ESPECIFICACAO DO CONTEXTO DA ROTA

(DEFUN ENTRA-CONTEXTO-ROTA (
                            LIN COL)
(SETQ PRE-CONTEXTO NIL
      STAT 'NAO)
(LE-ARQ        'CONFIG.MEM)(LE-ARQ        'ESCALA.MEM)(LE-ARQ
'DATABASE.MEM)
((NULL LSTCELL)(WRITE-STRING "A Representacao por celas  nao
existe!")(SYSTEM))
(IF (NEQ DEFINE-AREA 'OK)(MOTOR_DE_INFERENCIA))
(SETQ DRIVER 'ENTRA-CONTEXTO-ROTA)
(MAKE-WINDOW 20 0 5 80)
(CLEAR-SCREEN)
(DESCRICAO-DA-ROTA LBL-ROTA)
(IF (> LBL-ROTA 1)
 (PROGN
  (LIBERA-MEMORIA)
  (CLEAR-SCREEN)
  ((S-OU-N-P "DESEJA ASSOCIAR A ROTA  O  MESMO  CONTEXTO  DA
ROTA ANTERIOR? ")
   (SETQ STAT 'SIM)
   (RECUPERA-CONHEC-ANTERIOR LBL-ROTA (- LBL-ROTA 1) )) ))
(CLEAR-SCREEN)
(IF (NEQ DEFINE-AREA 'OK)
    (IF (S-OU-N-P "DESEJA INICIAR A DEFINICAO  DE  AREAS  DO
LAYOUT ? ")
        (ENTRADA-CONHEC-AREAS)))
(SETQ DEFINE-AREA 'OK)
(CLEAR-SCREEN)
(IF (EQ STAT 'NAO)
    (PROGN
      (ENTRADA-RESTRICOES)
      (INF-AREAS-DE-PREFERENCIA)) )
(CLEAR-SCREEN)
(IF (EQ STAT 'NAO)
```

```
    (PROGN
      (PROMPT-READ-CHAR '(C)
                "TECLE  PARA INICIAR A DEFINICAO DA   FUNCAO
DE AVALIACAO")
       (DEFINE-FUNCAO-GLOBAL  LBL-ROTA)))
(DEFINE-VIA-POINTS LBL-ROTA)
(GERA-DATABASE LBL-ROTA)
(MOTOR_DE_INFERENCIA)
(GERA-FRAME-ROTA CONTEXTO LBL-ROTA)
(DETERMINA-ROTA LBL-ROTA)
(EMISSAO-DE-RESULTADOS FRAME)
 )

(DEFUN LIBERA-MEMORIA ()
(SETQ WRS NIL)(TERPRI)
(WRITE-STRING "LIBERANDO MEMORIA...")(TERPRI)
(SETQ *FREE-LIST* (NCONC WF *FREE-LIST*)
       *FREE-LIST* (NCONC WB *FREE-LIST*)
       *FREE-LIST* (NCONC REGRAS *FREE-LIST*)
       *FREE-LIST* (NCONC PRE-CONTEXTO  *FREE-LIST*)
       *FREE-LIST* (NCONC LST-VARIAVEIS *FREE-LIST*)
       *FREE-LIST* (NCONC PRED-OBST *FREE-LIST*))
(ALLOCATE 4096)
(WRITE-STRING "FIM DA OPERACAO") )

(DEFUN FINALIZA ()
(SET-CURSOR 4 0)
(PROMPT-READ-CHAR  '(F)    "                                  <F>   -
FINALIZA"))

;ENTRADA DE INFORMACOES SOBRE AREAS

(DEFUN ENTRADA-CONHEC-AREAS (
                                  ROT LST RESP)
; Somente e executada uma vez pelo Sistema
; No Menu principal existem LST-AREAS e ATTRIB-AREAS

(LOOP
  (WRITE-STRING "DIGITE O ROTULO DA AREA  OU  TECLE  <F>  P/
FINALIZAR:")
  (SETQ ROT (RATOM))(TERPRI)(CLEAR-SCREEN)
  ((EQ ROT 'F))
 (WRITE-STRING  "DESCREVA  A  LISTA   DE   COORDENADAS   QUE
IDENTIFICAM A AREA:")
  (SET-CURSOR 4 0)
  (SETQ LST (READ))
  (SETQ LST (MAPCAN '(LAMBDA(X)(INSIDE-OBJECT X))LST))
  (PUSH (LIST ROT LST) LST-AREAS)
  (LOOP
    (CLEAR-SCREEN)
    (WRITE-STRING   "ENTRE   COM   O   PREDICADO   DA   AREA
:")(SET-CURSOR 1 0)
    (SETQ RESP (RATOM))
    (IF    (EQ     RESP     '?)(MOSTRA-OPCOES-AREAS     (GET
'DESCRITOR-DE-AREA 'BINDINGS))
        (PUSH (LIST RESP ROT) ATTRIB-AREAS))
    ((NEQ RESP '?)) ) ))
```

```
(DEFUN MOSTRA-OPCOES-AREAS (OPCOES
                                        CONTLIN)
(CLEAR-SCREEN)
(SETQ CONTLIN 0)
(LOOP
   ((NULL OPCOES)(CONTINUA-PROMPT))
   (COND
      ((EQ CONTLIN 2)(TERPRI)(CONTINUA-PROMPT)
                     (CLEAR-SCREEN)(SETQ CONTLIN 0)) )
   (PRINT (CAR OPCOES))(POP OPCOES)
   (INCQ CONTLIN) ) )

(DEFUN INSIDE-OBJECT (LST
                              BUSCA N)
((NULL LST)NIL)
(SETQ N 0)
(LOOP
(SETQ COORD (SECOND (NTH N LSTCELL)))
((NULL COORD))
((AND (< (CAAR COORD) (CAR LST) (CAADR COORD))
      (> (CADAR COORD) (SECOND LST) (CADR (THIRD COORD))))
      (SETQ BUSCA (CAR (NTH N LSTCELL))) )
(INCQ N)  )
(SETQ N 0)
(LOOP
((NOT (NULL BUSCA))BUSCA)
(SETQ COORD (CDR (NTH N OBSTACLE)))
((NULL COORD)NIL)
((AND (< (CAAR COORD) (CAR LST) (CAADR COORD))
      (> (CADAR COORD) (SECOND LST) (CADR (THIRD COORD))))
      (SETQ BUSCA (CAR (NTH N OBSTACLE))) )
(INCQ N)  )   )

(DEFUN CONTIDO-AREAS (ATOM LST)
(LOOP
((NULL LST))
((MEMBER ATOM (SECOND (POP LST))) ) ) )


; INICIO DO MODULO DE DESCRICAO DA ROTA


(DEFUN DESCRICAO-DA-ROTA (
                              DESCR RESP DESCR-I LST-TIPOW)
(WRITE-STRING "ROTA A SER DETERMINADA => ")(PRIN1 (PACK*  'R
(+ 1 LBL-ROTA)))
(TERPRI)
(INCQ LBL-ROTA)
(IF  (> LBL-ROTA 1)
     (IF (S-OU-N-P "A ROTA POSSUI AS MESMAS  PROPIEDADES  DA
ROTA ANTERIOR?")
         (PROGN
          (RECUPERA-DESCRICAO-ROTA LBL-ROTA (- LBL-ROTA 1))
          (RETURN) ) ) )
(SETQ ATTRIB-ROTA NIL)
(SETQ LST-TIPOW (GET 'DESCRITOR-DE-ROTA 'BINDINGS))
((NULL LST-TIPOW)(PRINT '(ERRO NA  ROTINA  DE  DESCRICAO  DE
ROTA)))
```

```
(PUSH (LIST 'ROTA (PACK* 'R LBL-ROTA)) ATTRIB-ROTA)
(LOOP
 ((NULL LST-TIPOW)(PRINT ATTRIB-ROTA)(FINALIZA))
 (IF (S-OU-N-P (CAAR LST-TIPOW))
     (PUSH (LIST (CAR(POP  LST-TIPOW))(PACK*  'R  LBL-ROTA))
ATTRIB-ROTA)
     (POP LST-TIPOW))
 (CLEAR-SCREEN)
  ) )


; INICIO ENTRADA DE INFORMACOES SOBRE RESTRICOES


(DEFUN ENTRADA-RESTRICOES (
                                LST-RESTR RESTR RESP RESTR-I)
(CLEAR-SCREEN)
(PROMPT-READ-CHAR '(C)
     "TECLE  PARA INICIAR A DEFINICAO DE   RESTRICOES")
(CLEAR-SCREEN)
(IF (> LBL-ROTA 1)
  (PROGN
   ((S-OU-N-P
      "DESEJA ASSOCIAR A ROTA AS MESMAS         RESTRICOES DA
ROTA ANTERIOR?")
    (RECUPERA-CONHEC-RESTRICOES LBL-ROTA (- LBL-ROTA 1))
   (RETURN)) ))
(SETQ ATTRIB-RESTR NIL)  ;Contem todas as clausulas  geradas
na fase de entrada
                         ;de restricoes
(SETQ LST-RESTR (GET 'RESTRICAO 'BINDINGS))
(LOOP
((NULL LST-RESTR)(CLEAR-SCREEN))
(SETQ RESTR-I (CAR(POP LST-RESTR)))
(SETQ RESTR (PACK* 'INF- RESTR-I))
(COND
((OR          (NULL          (GETD          RESTR-I))(NULL(GETD
RESTR)))(TERPRI)(WRITE-STRING
  "ERRO FATAL. NAO EXISTE A  INFORMACAO  DE  COMO  DEVE  SER
TRATADA A RESTRICAO ")
   (PRIN1  RESTR-I)(SETQ  RESP  (PROMPT-READ-CHAR  '(S  N)
"CONTINUA <S/N> "))
   (IF (OR(EQ RESP 'N)(EQ RESP 'n)) (SYSTEM)))
(T (EVAL (LIST RESTR))) ) ) )

(DEFUN INF-AREA-NAO-RECOMENDADA (LBL
                                      CLAUSULA          CLAUS-B
CLAUS-C)
;  FOI  TIRADA  DE  FATOS.MEM  A  DESCRICAO  DE  RESTRICAO
'AREA-NAO-RECOMENDADA
(CLEAR-SCREEN)
((NOT(S-OU-N-P "DESEJA  ENTRAR  COM  ROTULOS  DE  AREAS  NAO
RECOMENDADAS?")))
(PUSH 'AREA-NAO-RECOMENDADA RESTR-ROTA)
(LOOP
    (CLEAR-SCREEN)
    (WRITE-LINHA  '(AREA?)  0   0)(SPACES   10)(WRITE-STRING
"<F>-FINALIZA")
```

```
    (SET-CURSOR 0 9)
    (RATOM)
    ((EQ RATOM 'F))
    (SETQ CLAUSULA (LIST 'AREA-NAO-RECOMENDADA (PACK* 'R
LBL-ROTA) RATOM)
          CLAUS-B          (SUBSTITUTE          'AREA-PROIBIDA
'AREA-NAO-RECOMENDADA CLAUSULA)
          CLAUS-C       (SUBSTITUTE       'AREA-DE-PREFERENCIA
'AREA-NAO-RECOMENDADA
                                    CLAUSULA))
    (IF (AND(NULL(MEMBER CLAUSULA CONTEXTO 'EQUAL))
            (NULL(MEMBER CLAUS-B  CONTEXTO 'EQUAL))
            (NULL(MEMBER CLAUS-C  CONTEXTO 'EQUAL)))
        (PUSH CLAUSULA ATTRIB-AREAS)
        (WRITE-STRING "NAO E PERMITIDO. FOI EVITADO CONFLITO
NA BC")) ) )

(DEFUN INF-AREA-PROIBIDA (LBL
                                    CLAUSULA)
(CLEAR-SCREEN)
((NOT(S-OU-N-P  "DESEJA  ENTRAR  COM  ROTULOS  DE  AREAS
PROIBIDAS?")))
(PUSH 'AREA-PROIBIDA RESTR-ROTA)
(LOOP
    (CLEAR-SCREEN)
    (WRITE-LINHA  '(AREA?)  0   0)(SPACES   10)(WRITE-STRING
"<F>-FINALIZA")
    (SET-CURSOR 0 9)
    (RATOM)
    ((EQ RATOM 'F))
    (SETQ CLAUSULA (LIST 'AREA-PROIBIDA (PACK* 'R  LBL-ROTA)
RATOM)
          CLAUS-B     (SUBSTITUTE       'AREA-NAO-RECOMENDADA
'AREA-PROIBIDA CLAUSULA)
          CLAUS-C       (SUBSTITUTE       'AREA-DE-PREFERENCIA
'AREA-PROIBIDA
                                    CLAUSULA))
    (IF (AND(NULL(MEMBER CLAUSULA CONTEXTO 'EQUAL))
            (NULL(MEMBER CLAUS-B  CONTEXTO 'EQUAL))
            (NULL(MEMBER CLAUS-C  CONTEXTO 'EQUAL)))
        (PUSH CLAUSULA ATTRIB-AREAS)
        (WRITE-STRING "NAO E PERMITIDO. FOI EVITADO CONFLITO
NA BC")) ) )

(DEFUN INF-AREAS-DE-PREFERENCIA (LBL
                                        CLAUSULA CLAUS-B CLAUS-C)
(SETQ ATTRIB-HEUR NIL)
(CLEAR-SCREEN)
((NOT(S-OU-N-P  "DESEJA  ENTRAR  COM  ROTULOS  DE  AREAS  DE
PREFERENCIA?")))
(LOOP
    (CLEAR-SCREEN)
    (WRITE-LINHA  '(AREA?)  0   0)(SPACES   10)(WRITE-STRING
"<F>-FINALIZA")
    (SET-CURSOR 0 9)
    (RATOM)
    ((EQ RATOM 'F))
    (SETQ  CLAUSULA  (LIST  'AREA-DE-PREFERENCIA  (PACK*  'R
```

```
LBL-ROTA) RATOM)
          CLAUS-B          (SUBSTITUTE          'AREA-PROIBIDA
'AREA-DE-PREFERENCIA CLAUSULA)
          CLAUS-C      (SUBSTITUTE       'AREA-NAO-RECOMENDADA
'AREA-DE-PREFERENCIA
                                  CLAUSULA))
    (IF (AND(NULL(MEMBER CLAUSULA CONTEXTO 'EQUAL))
            (NULL(MEMBER CLAUS-B  CONTEXTO 'EQUAL))
            (NULL(MEMBER CLAUS-C  CONTEXTO 'EQUAL)))
        (PUSH CLAUSULA ATTRIB-HEUR)
        (WRITE-STRING "NAO E PERMITIDO. FOI EVITADO CONFLITO
NA BC")) ) )


(DEFUN INF-LARGURA-MINIMA-ROTA ()
(CLEAR-SCREEN)
((S-OU-N-P "A LARGURA DA ROTA E RELEVANTE?")
  (WRITE-STRING "LARGURA DA ROTA? ")
  (PUSH  (LIST  'LARGURA-MINIMA-ROTA  (PACK*  'R   LBL-ROTA)
(RATOM)) ATTRIB-RESTR)
  (PUSH 'LARGURA-MINIMA-ROTA RESTR-ROTA))
(PUSH (LIST  'LARGURA-MINIMA-ROTA  (PACK*  'R  LBL-ROTA)  0)
ATTRIB-RESTR)
(PUSH 'LARGURA-MINIMA-ROTA RESTR-ROTA) )

(DEFUN INF-SEGMENTO-MINIMO ()
(CLEAR-SCREEN)
((S-OU-N-P
    "E NECESSARIO QUE OS  SEGMENTOS  DE  ROTA    POSSUAM  UM
CUMPRIMENTO MINIMO?")
  (TERPRI)(WRITE-STRING "CUMPRIMENTO MINIMO = ")
  (PUSH (LIST 'SEGMENTO-MINIMO (PACK*  'R  LBL-ROTA)(RATOM))
ATTRIB-RESTR)
  (PUSH 'SEGMENTO-MINIMO RESTR-ROTA)) )

(DEFUN INF-DISTANCIA-MINIMA-RECOMENDADA (LBL
                                                DISTANCIA OBST)
(CLEAR-SCREEN)
((S-OU-N-P  "DESEJA  ENTRAR  COM  VALORES  ESPECIFICOS  DE
ESPACAMENTO RECOMENDADO?")
  (PUSH 'DISTANCIA-MINIMA-RECOMENDADA RESTR-ROTA)
  (LOOP
     (CLEAR-SCREEN)
     (WRITE-LINHA '(OBSTACULO ?) 0 1)(SET-CURSOR 4 22)
     (WRITE-STRING "<F>-FINALIZA")
     (SET-CURSOR 0 14)
     (SETQ OBST (RATOM))
     ((EQ RATOM 'F))
     (WRITE-LINHA  '(DISTANCIA  :)   1   1)(SETQ   DISTANCIA
(RATOM))
     (PUSH  (LIST  'DISTANCIA-MINIMA-RECOMENDADA  (PACK*  'R
LBL-ROTA) OBST
           DISTANCIA) ATTRIB-RESTR)  ) )  )

(DEFUN INF-DISTANCIA-MINIMA (LBL
                                  OBST DISTANCIA)
(CLEAR-SCREEN)
((S-OU-N-P  "DESEJA  ENTRAR  COM  VALORES  ESPECIFICOS  DE
```

```
ESPACAMENTO MINIMO?")
 (PUSH 'DISTANCIA-MINIMA RESTR-ROTA)
 (LOOP
    (CLEAR-SCREEN)
    (WRITE-LINHA '(OBSTACULO?) 0 1)
    (SET-CURSOR 4 22)
    (WRITE-STRING "<F> - FINALIZA")
    (SET-CURSOR 0 14)
    (SETQ OBST (RATOM))
    ((EQ RATOM 'F))
    (WRITE-LINHA '(DISTANCIA :) 1 1)(SETQ DISTANCIA (RATOM))
    (PUSH (LIST 'DISTANCIA-MINIMA (PACK* 'R LBL-ROTA) OBST
DISTANCIA)
          ATTRIB-RESTR)) ) )

(DEFUN INF-VIZINHANCA-PROIBIDA (LBL)
; FOI TIRADA DE FATOS.MEM A DESCRICAO DE RESTRICAO
VIZINHANCA-NAO-RECOMENDADA
(CLEAR-SCREEN)
((S-OU-N-P
  "DESEJA ESPECIFICAR OS OBSTACULOS QUE REPRESENTAM
VIZINHANCA PROIBIDA?")
  (PUSH 'VIZINHANCA-PROIBIDA RESTR-ROTA)
  (TERPRI)
  (LOOP
    (CLEAR-SCREEN)
    (WRITE-LINHA '(OBSTACULO?) 0 1)(SET-CURSOR 4 22)
    (WRITE-STRING "<F> - FINALIZA")
    (SET-CURSOR 0 14)
    (RATOM)
    ((EQ RATOM 'F))
    (PUSH(LIST 'VIZINHANCA-PROIBIDA (PACK* 'R LBL-ROTA)
RATOM)ATTRIB-RESTR))))

(DEFUN INF-VIZINHANCA-NAO-RECOMENDADA (LBL)
; FOI TIRADA DE FATOS.MEM A DESCRICAO DE RESTRICAO
VIZINHANCA-NAO-RECOMENDADA
(CLEAR-SCREEN)
((S-OU-N-P
  "DESEJA ESPECIFICAR OS OBSTACULOS QUE REPRESENTAM
VIZINHANCA NAO RECOMENDADA?")
  (PUSH 'VIZINHANCA-NAO-RECOMENDADA RESTR-ROTA)
  (TERPRI)
  (LOOP
    (CLEAR-SCREEN)
    (WRITE-LINHA '(OBSTACULO?) 0 1)(SET-CURSOR 4 22)
    (WRITE-STRING "<F> - FINALIZA")
    (SET-CURSOR 0 14)
    (RATOM)
    ((EQ RATOM 'F))
    (PUSH(LIST 'VIZINHANCA-NAO-RECOMENDADA (PACK* 'R
LBL-ROTA) RATOM)
                ATTRIB-RESTR))))

(DEFUN INF-DIRECAO-PROIBIDA (LBL
                             OPCOES)
(CLEAR-SCREEN)
((S-OU-N-P "DESEJA ESPECIFICAR UMA DIRECAO PROIBIDA PARA A
```

```
ROTA?")
  (TERPRI)
  (SETQ OPCOES '((N NORTE)(L LESTE)(S SUL)(O OESTE)))
  (PUSH (LIST 'DIRECAO-PROIBIDA (PACK* 'R LBL-ROTA)
    (SECOND (ASSOC (PROMPT-READ-CHAR '(N S L O)
   "<N> - NORTE  <L> - LESTE  <S> - SUL     <O> -  OESTE   -
QUAL ?") OPCOES)))
        ATTRIB-RESTR)
  (PUSH 'DIRECAO-PROIBIDA RESTR-ROTA) ) )

(DEFUN INF-DIRECAO-NAO-RECOMENDADA (LBL
                                    OPCOES)
(CLEAR-SCREEN)
((S-OU-N-P  "DESEJA    ESPECIFICAR    UMA    DIRECAO    NAO
RECOMENDADA PARA A ROTA?")
 (TERPRI)
  (SETQ OPCOES '((N NORTE)(L LESTE)(S SUL)(O OESTE)))
  (PUSH (LIST 'DIRECAO-NAO-RECOMENDADA (PACK* 'R LBL-ROTA)
        (ASSOC OPCOES (PROMPT-READ-CHAR '(N S L O)
        "<N> - NORTE  <L> - LESTE  <S> - SUL     <O> - OESTE
- QUAL ?")))
        ATTRIB-RESTR)
  (PUSH 'DIRECAO-NAO-RECOMENDADA RESTR-ROTA) ) )

(DEFUN INF-CRUZAMENTO-PROIBIDO (LBL)
(CLEAR-SCREEN)
((S-OU-N-P "OS CRUZAMENTOS DE ROTAS  SAO  PROIBIDOS    NESTE
CASO?")
 (PUSH  (LIST  'CRUZAMENTO-PROIBIDO  (PACK*  'R   LBL-ROTA))
ATTRIB-RESTR)
 (PUSH 'CRUZAMENTO-PROIBIDO RESTR-ROTA)) )

(DEFUN INF-COINCIDENCIA-PROIBIDA (LBL)
(CLEAR-SCREEN)
((S-OU-N-P "A COINCIDENCIA  PARCIAL  OU  TOTAL  DE  ROTAS  E
PROIBIDA?")
 (PUSH (LIST  'COINCIDENCIA-PROIBIDA  (PACK*  'R  LBL-ROTA))
ATTRIB-RESTR)
 (PUSH 'COINCIDENCIA-PROIBIDA RESTR-ROTA)) )

(DEFUN RECUPERA-CONHEC-RESTRICOES (ATUAL ANTERIOR)
(SETQ ATTRIB-RESTR (MAPCAN
'(LAMBDA(X)(SUBSTITUTE (PACK* 'R ATUAL)(PACK*  'R  ANTERIOR)
X)) ATTRIB-RESTR))
(SETQ RESTR-ROTA (MAPCAN
'(LAMBDA(X)(SUBSTITUTE (PACK* 'R ATUAL) (PACK* 'R  ANTERIOR)
X)) RESTR-ROTA))
(SETQ ATTRIB-AREAS (MAPCAN
'(LAMBDA(X)(SUBSTITUTE (PACK* 'R ATUAL)(PACK*  'R  ANTERIOR)
X)) ATTRIB-AREAS))
(WRITE-STRING "ATRIBUICAO DE RESTRICOES PARA A NOVA     ROTA
COMPLETADA")
(CONTINUA-PROMPT) )

(DEFUN DEFINE-FUNCAO-GLOBAL (LBL
                             NUM-COMP FUNCAO-LISTW MEM NW)
(CLEAR-SCREEN)
(IF (> LBL-ROTA 1)
```

```
    (IF (S-OU-N-P
     "DESEJA CONTINUAR UTILIZANDO A FUNCAO GLOBAL DE
AVALIACAO CORRENTE?")
      (PROGN
        (RECUPERA-FUNCAO-GLOBAL LBL-ROTA (- LBL-ROTA 1))
        (RETURN) ) ) )
(SETQ FUNCAO-LIST NIL)
(WRITE-STRING "QUANTOS COMPONENTES POSSUE A FUNCAO
GLOBAL DE AVALIACAO ?")
(SETQ NUM-COMP (RATOM))(CLEAR-SCREEN)(SET-CURSOR 0 1)
(WRITE-STRING "Estrutura de cada componente :          (Ccl
(Cpg g)(Cph h))
 onde :")(TERPRI)(CONTINUA-PROMPT)
(WRITE-STRING "Ccl = Valor do coef. de comb. linear do
componente")(TERPRI)
(CONTINUA-PROMPT)
(WRITE-STRING "Cpg = Valor do coef. de ponderacao de
g")(TERPRI)
(WRITE-STRING "g   = Rotulo de indentif. da funcao g")
(TERPRI)(CONTINUA-PROMPT)
(WRITE-STRING "Cph = Valor do coef. de ponderacao de
h")(TERPRI)
(WRITE-STRING "h = Rotulo de identificacao da funcao h")
(TERPRI)(CONTINUA-PROMPT)
(WRITE-STRING "Caso nao exista a funcao h ,")
(TERPRI)
(WRITE-STRING "a digitacao do par (Cph h) sera
omitida")(TERPRI)
(CONTINUA-PROMPT)
(SETQ FUNCAO-LINK (GET 'EQUIVALENCIA-FUNCAO-MANIPULAVEL
'BINDINGS))
(SETQ NW 1)
(LOOP
((> NW NUM-COMP)
 (PROGN
   (CLEAR-SCREEN)(WRITE-LINHA '(FUNCAO GLOBAL =) 0  1)(PRIN1
FUNCAO-LIST)
   (TERPRI)(CONTINUA-PROMPT)
   (LOOP
     (SETQ MEM (POP FUNCAO-LIST))
     (IF (NOT(NULL (THIRD MEM)))
        (PUSH (LIST(LIST (* (CAR MEM) (CAADR MEM))
(SECOND(ASSOC (CADADR MEM)
         FUNCAO-LINK))) (LIST (* (CAR MEM)(CAADDR MEM))
          (SECOND(ASSOC (SECOND(THIRD MEM)) FUNCAO-LINK))))
FUNCAO-LISTW)
        (PUSH (LIST(LIST (* (CAR MEM)(CAADR MEM))
          (SECOND (ASSOC (CADADR MEM) FUNCAO-LINK)) ))
FUNCAO-LISTW))
     ((NULL FUNCAO-LIST)(SETQ FUNCAO-LIST FUNCAO-LISTW)) ) )
)
(CLEAR-SCREEN)
(WRITE-LINHA (APPEND '(ESTRUTURA DO COMPONENTE) (LIST NW)) 0
0)
(SPACES 1)(PRINC ':)
(WRITE-LINHA '((?) - MOSTRA OPCOES DE FUNCOES G E H) 3 0)
(SET-CURSOR 1 0)
(SETQ MEMBRO (READ))
```

```
(IF (EQUAL MEMBRO '(?))(MOSTRA-OPCOES-G-H FUNCAO-LINK)
    (PROGN
       (PUSH MEMBRO FUNCAO-LIST)
       (INCQ NW) ))
(CLEAR-SCREEN) ) )

(DEFUN MOSTRA-OPCOES-G-H (FUNCOES-W
                          CONTLIN)
(CLEAR-SCREEN)
(SETQ CONTLIN 0)
(LOOP
  ((NULL FUNCOES-W)(CONTINUA-PROMPT))
  (COND
     ((EQ CONTLIN 2)(TERPRI)(CONTINUA-PROMPT)
                    (CLEAR-SCREEN)
                    (SETQ CONTLIN 0) ) )
  (PRINT (CAR FUNCOES-W))(POP FUNCOES-W)
  (INCQ CONTLIN) ) )

(DEFUN CONTINUA-PROMPT ()
(SET-CURSOR 4 24)
          (PROMPT-READ-CHAR '(C) " - CONTINUA ")
(CLEAR-SCREEN) )


(DEFUN RECUPERA-CONHEC-ANTERIOR (LBL-AT LBL-ANT)
(RECUPERA-FUNCAO-GLOBAL LBL-AT LBL-ANT)
(RECUPERA-CONHEC-RESTRICOES LBL-AT LBL-ANT)
(RECUPERA-AREAS-PREFERENCIA LBL-AT LBL-ANT)
(WRITE-STRING "ATRIB DE FUNC DE AVAL, RESTRS E HEUR.
COMPLETA")
(CONTINUA-PROMPT) )

(DEFUN RECUPERA-AREAS-PREFERENCIA (ATUAL ANTERIOR)
(SETQ ATTRIB-HEUR (MAPCAN '(LAMBDA(X)(SUBSTITUTE (PACK* 'R
ATUAL)
      (PACK* 'R ANTERIOR) X)) ATTRIB-HEUR)) )

(DEFUN RECUPERA-DESCRICAO-ROTA (ATUAL ANTERIOR)
(SETQ ATTRIB-ROTA (MAPCAN
'(LAMBDA(X)(SUBSTITUTE (PACK* 'R ATUAL) (PACK* 'R ANTERIOR)
X)) ATTRIB-ROTA))
(WRITE-STRING "ELABORACAO DA DESCRICAO DA ROTA COMPLETADA")
(CONTINUA-PROMPT) )

(DEFUN RECUPERA-FUNCAO-GLOBAL (ATUAL ANTERIOR)
(SETQ FUNCAO-LIST (MAPCAN
'(LAMBDA(X)(SUBSTITUTE (PACK* 'R ATUAL) (PACK* 'R ANTERIOR)
X)) FUNCAO-LIST))
(WRITE-STRING "CONFECCAO DA FUNCAO GLOBAL DE AVALIACAO
COMPLETADA")
(PRINT FUNCAO-LIST)
(CONTINUA-PROMPT) )

(DEFUN DEFINE-VIA-POINTS (LBL-R
                          X Y TEXTO)
(CLEAR-SCREEN)
(SETQ LIST-VIA-POINTS NIL)
```

```
(SETQ TEXTO '((OS VIA-POINTS SAO PONTOS POR ONDE A ROTA)
              (OBRIGATORIAMENTE DEVE PASSAR)
              (O SISTEMA TRANSFORMA ESTA INFORMACAO)
              (EM CELAS POR ONDE A ROTA DEVERA)
              (PASSAR VOCE DEVERA INFORMAR QUAIS)
              (SAO ESTES PONTOS E A SEQUENCIA EM)
              (QUE ESTES SERAO PERCORRIDOS)))
((S-OU-N-P "DESEJA ESPECIFICAR VIA-POINTS PARA A    ROTA")
  (LOOP
     (CLEAR-SCREEN)
     (WRITE-LINHA '(X) 0 11)(WRITE-LINHA '(Y) 0 16)
     (WRITE-LINHA '(VIA POINT) 1 0)(SET-CURSOR 1 11)(SETQ  X
(RATOM))
     (SET-CURSOR 1 16)(SETQ Y (RATOM))
     (SETQ  LIST-VIA-POINTS  (APPEND  LIST-VIA-POINTS  (LIST
(LIST X Y))))
     (TERPRI)(SET-CURSOR 4 0)
     ((NOT(S-OU-N-P "                OUTRO VIA-POINT ?")))   )
) )

(DEFUN GERA-DATABASE (LBL-R)
(SETQ PRE-CONTEXTO NIL)
(IF   (NOT(NULL   PRED-OBST))(SETQ   PRE-CONTEXTO    (APPEND
PRE-CONTEXTO PRED-OBST)))
(IF (NOT(NULL ATTRIB-AREAS))
    (SETQ PRE-CONTEXTO (APPEND PRE-CONTEXTO ATTRIB-AREAS)))
(IF (NOT(NULL ATTRIB-RESTR))
                    (SETQ  PRE-CONTEXTO  (APPEND  PRE-CONTEXTO
ATTRIB-RESTR)))
(IF (NOT(NULL ATTRIB-HEUR))
                    (SETQ  PRE-CONTEXTO  (APPEND  PRE-CONTEXTO
ATTRIB-HEUR)))
(IF (NOT(NULL ATTRIB-ROTA))
                    (SETQ  PRE-CONTEXTO  (APPEND  PRE-CONTEXTO
ATTRIB-ROTA))) )


(PROGN
((S-OU-N-P    "O    PROGRAMA    ESTA    OK    ?")(LE-ARQ
'ROUTER2.LSP)(LE-ARQ 'ROUTER3.LSP)
(LE-ARQ 'INFERENC.LSP)(LE-ARQ 'GRAPH.LSP)
(LE-ARQ 'FILEIO.LSP)(LE-ARQ 'EXTENDL.LSP)(LE-ARQ 'DEBUG.LSP)
(SETQ LBL-ROTA 0
      ATTRIB-AREAS NIL       ATTRIB-RESTR NIL
      ATTRIB-ROTA  NIL       LST-AREAS    NIL
      DEFINE-AREA  NIL
      RESTR-ROTA   NIL)
(SETQ DRIVER 'ENTRA-CONTEXTO-ROTA)
(WRITE-STRING  "PARA  SALVAR  O  ESTADO  ATUAL  DO  PROGRAMA
DIGITAR 'SAVE ROUTER2'")
(TERPRI 2)
(RDS)) )

(RETURN)
```

A.1.8 MODULO ROUTER2.LSP.

```
(DEFUN GERA-FRAME-ROTA (FRM-CONTEXTO LBL-R
                                                    VARW)
(SETQ FRAME NIL
      FRAME (APPEND FRAME (GERA-PREENCHE-SLOT-ROTULO LBL-R))
      FRAME   (APPEND   FRAME    (GERA-PREENCHE-SLOT-INICIAL
LBL-R))
      FRAME   (APPEND   FRAME   (GERA-PREENCHE-SLOT-OBJETIVO
LBL-R))
      FRAME (APPEND FRAME (GERA-PREENCHE-SLOT-TIPO LBL-R))
      FRAME  (APPEND  FRAME   (GERA-PREENCHE-SLOT-VIA-POINTS
LBL-R))
      FRAME (APPEND FRAME (GERA-SLOT-PAI-DE LBL-R))
      FRAME (APPEND FRAME (GERA-SLOT-PARTE-DE LBL-R))
      FRAME  (APPEND  FRAME  (GERA-SLOT-AREAS-DE-PREFERENCIA
LBL-R))
      FRAME (APPEND FRAME (GERA-SLOTS-RESTRICOES LBL-R))
      FRAME (APPEND FRAME  (GERA-PREENCHE-SLOT-FUNCAO-GLOBAL
LBL-R))
      FRAME   (APPEND   FRAME    (GERA-PREENCHE-SLOT-SOLUCAO
LBL-R)))  )
;       FRAME  (APPEND   FRAME   (GERA-PREENCHE-SLOT-LARGURA
LBL-R))


(DEFUN GERA-PREENCHE-SLOT-ROTULO (LBL)
(LIST(LIST 'ROTULO (PACK* 'R LBL))) )

(DEFUN GERA-PREENCHE-SLOT-TIPO (LBL)
(LIST(LIST 'TIPO ATTRIB-ROTA)) )

;(DEFUN GERA-PREENCHE-SLOT-LARGURA (LBL)
;(LIST(LIST 'LARGURA (CAR(GET 'LARGURA-DA-ROTA 'BINDINGS))))
)

(DEFUN GERA-PREENCHE-SLOT-INICIAL (LBL)
(LIST(LIST       'INICIAL       (INSIDE-OBJECT-B         (CAR
LIST-VIA-POINTS)))) )

(DEFUN GERA-PREENCHE-SLOT-OBJETIVO (LBL)
(LIST(LIST     'OBJETIVO     (INSIDE-OBJECT-B      (CAR(LAST
LIST-VIA-POINTS))))) )

(DEFUN GERA-SLOT-PAI-DE (LBL)
(LIST(LIST 'PAI-DE NIL)) )

(DEFUN GERA-SLOT-PARTE-DE (LBL)
(LIST(LIST 'PARTE-DE NIL)) )

(DEFUN GERA-SLOT-AREAS-DE-PREFERENCIA (LBL)
(LIST(LIST 'AREAS-DE-PREFERENCIA (CAR(MAPCAN '(LAMBDA(X)
            (SECOND(ASSOC (SECOND X) LST-AREAS)))
     (PARCIAL-MATCH (PACK* 'R LBL)(GET  'AREA-DE-PREFERENCIA
'BINDINGS))))) ) )

(DEFUN GERA-SLOTS-RESTRICOES (LBL
```

```
                                        N  RESTR-ROTA-W  SLOTS-RESTR
BIND-TEST)
(SETQ RESTR-ROTA-W (MAPCAR 'CAR (GET 'RESTRICAO 'BINDINGS))
      N 0)
(LOOP
  ((NULL (NTH N RESTR-ROTA-W))(APPEND(LIST(LIST  'RESTRICOES
RESTR-ROTA-W))
                        SLOTS-RESTR) )
  (IF (NULL(GET (NTH N RESTR-ROTA-W) 'BINDINGS))
      (DELETE (NTH N RESTR-ROTA-W) RESTR-ROTA-W)
      (PROGN
        (SETQ BIND-TEST
       (PARCIAL-MATCH   (PACK*   'R   LBL)   (GET   (NTH   N
RESTR-ROTA-W) 'BINDINGS)))
        (IF (NULL BIND-TEST) (DELETE  (NTH  N  RESTR-ROTA-W)
RESTR-ROTA-W)
        (PROGN
        (SETQ  SLOTS-RESTR  (APPEND  SLOTS-RESTR  (LIST(LIST
(NTH N RESTR-ROTA-W)
                                    BIND-TEST))) )
        (INCQ N) )) ) )   ) )

(DEFUN GERA-PREENCHE-SLOT-FUNCAO-GLOBAL (LBL)
(LIST(LIST 'FUNCAO-GLOBAL-DE-AVALIACAO FUNCAO-LIST)) )

(DEFUN GERA-PREENCHE-SLOT-SOLUCAO (LBL)
(LIST(LIST 'SOLUCAO NIL)) )

(DEFUN GERA-PREENCHE-SLOT-VIA-POINTS (LBL
                                        VIA-POINTS-LST
ARGM)
(LOOP
  ((NULL     LIST-VIA-POINTS)(LIST     (LIST     'VIA-POINTS
VIA-POINTS-LST)))
  (SETQ ARGM (POP LIST-VIA-POINTS))
  (SETQ      VIA-POINTS-LST      (APPEND      VIA-POINTS-LST
(LIST(INSIDE-OBJECT-B ARGM))))
          ) )

(DEFUN INSIDE-OBJECT-B (LST
                        BUSCA N COORD)
((NULL LST)NIL)
(SETQ N 0)
(LOOP
  ((NULL (NTH N LSTCELL))(PRIN1 'ERRO)(SYSTEM))
  (SETQ COORD (SECOND (NTH N LSTCELL)))
  ((AND (<= (CAAR COORD) (CAR LST) (CAADR COORD))
        (>=  (CADAR  COORD)  (SECOND  LST)  (CADR  (THIRD
COORD))))
        (SETQ BUSCA (CAR(NTH N LSTCELL))) )
  (INCQ N) ) )


(RDS)
```

A.1.9 MODULO ROUTER3.LSP.

```
(DEFUN DETERMINA-ROTA (LBL-R
                                    SUB-FRAMES LST)
(SETQ WF-G 0 WB-G 0  LST-VARIAVEIS NIL)
(IF  (NULL(SECOND(ASSOC 'VIA-POINTS FRAME)))
     (SETQ FRAME (DETERMINA-SUBROTA FRAME))
     (PROGN
        (SETQ SUB-FRAMES (GERA-SUBFRAMES FRAME))
        (LOOP
           ((NULL SUB-FRAMES))
           (SETQ     FRAME       (TRANSFERE-SOLUCAO       FRAME
(DETERMINA-SUBROTA
                          (POP SUB-FRAMES)) )) ) ) )
(PUSH FRAME LIST-FRAMES) )

(DEFUN GERA-SUBFRAMES (FRAME
                                   LST    LST-FRM    FRM-FILHO    CONT
INICIAL-F)
(SETQ LST (SECOND(ASSOC  'VIA-POINTS FRAME))
      CONT 1)
(LOOP
   ((NULL (SECOND LST))LST-FRM)
   (SETQ INICIAL-F (POP LST))
   (SETQ FRM-FILHO (LIST (APPEND (LIST
         (LIST 'ROTULO  (PACK  (LIST  (SECOND(ASSOC  'ROTULO
FRAME))
            '-F)))
         (LIST 'TIPO (SECOND(ASSOC 'TIPO FRAME)))
         (LIST 'INICIAL INICIAL-F)
         (LIST 'OBJETIVO (CAR LST))
         (LIST 'PAI-DE NIL)
         (LIST 'PARTE-DE (SECOND(ASSOC 'ROTULO FRAME)) )
         (CDR(CDR(CDR(CDR(CDR(CDR(CDR FRAME))))))) )))
    (SETQ LST-FRM (APPEND LST-FRM FRM-FILHO)) )  )
      ;        (LIST      'LARGURA-DA-ROTA       (SECOND(ASSOC
'LARGURA-DA-ROTA FRAME)))

(DEFUN TRANSFERE-SOLUCAO (FRAME LST
                                        SOL-ANT SOL)
(SETQ SOL-ANT (SECOND(ASSOC 'SOLUCAO FRAME)))
(SETQ SOL (SECOND(ASSOC 'SOLUCAO LST)))
;(DELETE (CAR(LAST FRAME)) FRAME 'EQUAL)
(IF (EQ  SOL-ANT  NIL)(APPEND  FRAME  (LIST  (LIST  'SOLUCAO
SOL)))(APPEND FRAME
(LIST(LIST 'SOLUCAO (APPEND SOL (CDR SOL-ANT)))))  ) )

(DEFUN DETERMINA-SUBROTA (FRAME-R
                                  I NEIGHBORS CICLOA CICLOB)
(SETQ WF NIL N  NIL E NIL WB NIL LST-VARIAVEIS NIL)
(SETQ WF (APPEND WF (LIST(CALCULA-CUSTOS (FINDCELL
   (SECOND(ASSOC 'INICIAL FRAME-R))) NIL FRAME-R) )))
(SETQ CICLOA 1)
(LOOP
   ((NULL  WF)(CLEAR-SCREEN)(WRITE-STRING  "NENHUMA  CONEXAO
POSSIVEL PARA : ")
              (PRINT (SECOND(ASSOC 'ROTULO FRAME-R)))
```

```
                (SYSTEM))
    (SETQ E  (ACHA-MENOR-CELA WF)
          WF (REMOVE-CELA WF E)
          WB (APPEND WB (LIST E)))
    ((EQ (CAR E) (SECOND(ASSOC 'OBJETIVO FRAME-R)))
         (DELETE '(SOLUCAO NIL) FRAME-R 'EQUAL)
         (SETQ WF-G (+ (LENGTH WF) WF-G)
               WB-G (+ (LENGTH WB) WB-G))
         (APPEND      FRAME-R       (LIST(LIST       'SOLUCAO
(RECUPERA-CAMINHO E
                (CAR(SECOND E))) )) ) )
(SETQ NEIGHBORS (THIRD (FINDCELL (CAR E))))
(SETQ NEIGHBORS (REMOVE (CAR(SECOND E)) NEIGHBORS 'EQUAL))
(SETQ CICLOB 1)
(LOOP
  ((NULL NEIGHBORS))
  (SETQ N (POP NEIGHBORS))
  (SETQ VARIAVEIS (AVALIACAO-DE-RESTRICOES  FRAME-R  N  (CAR
E)))
;  (IF (EQ (SECOND(ASSOC 'VIAVEL VARIAVEIS)) 'NAO)
;      (PROGN
;       (WRITE-STRING "  N  NAO  E  VIAVEL")(PRIN1  'N)(PRINT
N)(TERPRI)
;       (WRITE-STRING " CELA PAI ")(PRIN1 E)(RATOM) ))
  (IF (EQ (SECOND(ASSOC 'VIAVEL VARIAVEIS)) 'SIM)
         (PROGN
           (SETQ N (CALCULA-CUSTOS (FINDCELL N) E FRAME-R))
           (COND
             ((NOT(NULL (SECOND(ASSOC (CAR N) WF))))
                (SETQ NB (LIST (CAR N) (SECOND(ASSOC (CAR N)
WF)))))
                (IF      (<       (CAR       (LAST       (SECOND
N)))(CAR(LAST(SECOND NB))))
                  (PROGN
                  (SETQ NB (REMOVE NB WF 'EQUAL))
                  (SETQ   LST-VARIAVEIS   (REMOVE   VARIAVEIS
LST-VARIAVEIS 'EQUAL))
                  (PUSH N WF)
                  (PUSH     (LIST(CAR     N)(CDR     VARIAVEIS))
LST-VARIAVEIS) ))
               ((NOT(NULL (SECOND(ASSOC (CAR N) WB))))
                  (SETQ NB (LIST (CAR N)  (SECOND(ASSOC  (CAR
N) WB))))
                  (IF      (<      (CAR       (LAST       (SECOND
N)))(CAR(LAST(SECOND NB))))
                  (PROGN
                  (SETQ NB (REMOVE NB WB 'EQUAL))
                  (SETQ   LST-VARIAVEIS   (REMOVE   VARIAVEIS
LST-VARIAVEIS 'EQUAL))
                  (PUSH N WF)
                  (PUSH     (LIST(CAR     N)(CDR     VARIAVEIS))
LST-VARIAVEIS) ))
                (T (PUSH N WF)
                  (PUSH   (LIST   (CAR   N)(CDR   VARIAVEIS))
LST-VARIAVEIS) ))))
(INCQ CICLOB      )(INCQ CICLOA)) )

(DEFUN ACHA-MENOR-CELA (WF-IMG
```

```
                                        MENOR)
(SETQ MENOR (POP WF-IMG))
(LOOP
((NULL WF-IMG)MENOR)
(IF (< (CAR(LAST(SECOND (CAR WF-IMG))))
    (CAR(LAST(SECOND MENOR))))
    (SETQ MENOR (CAR WF-IMG)) )
    (POP WF-IMG) ) )

(DEFUN REMOVE-CELA (LST1 CELA
                                    LST2 CELB)
((NULL LST1)NIL)
(SETQ CELB (CAR LST1))
((EQ (CAR CELA)(CAR CELB))(CDR LST1))
(CONS CELB (REMOVE-CELA (CDR LST1) CELA)) )


(DEFUN AVALIACAO-DE-RESTRICOES (FRM-IMG NCEL-IMG ECEL-IMG
                                        STATCEL    LST-RESTR
VARIAVEIS RETORNO
                                        RESTR-CORRENTE  ROTA
RESP E-PARM-VARS
                                        E-PARM)
;      EX:      VARIAVEIS      ->      ((VIAVEL      SIM)(SUCCESS
(LARGURA-MINIMA-ROTA)))
((NOT(ATOM NCEL-IMG))NIL)
(SETQ STATCEL (LIST 'STATUS 'VIAVEL)
      LST-RESTR (SECOND(ASSOC 'RESTRICOES FRM-IMG)))
(LOOP
(SETQ N-PARM (FINDCELL NCEL-IMG)
      E-PARM-VARS (SECOND(ASSOC ECEL-IMG LST-VARIAVEIS))
      E-PARM (FINDCELL ECEL-IMG))
  ((AND (NULL LST-RESTR)(EQUAL STATCEL '(STATUS VIAVEL)))
                (SETQ VARIAVEIS (CONS '(VIAVEL SIM)
                      (MAPCAN      '(LAMBDA(X)(CAR(CDR      X)))
VARIAVEIS) )))
  ((NULL LST-RESTR))
  (SETQ RESTR-CORRENTE (POP LST-RESTR)
        RETORNO    (EVAL    (LIST    RESTR-CORRENTE     'FRM-IMG
'E-PARM-VARS 'N-PARM
                           'E-PARM)))
  ((AND (EQ (CAR RETORNO) 'FAIL)
        (MEMBERP RESTR-CORRENTE (MAPCAN '(LAMBDA(X)(CAR X))
                  (GET 'RIGIDO 'BINDINGS))))
        (SETQ VARIAVEIS (CONS RETORNO VARIAVEIS))
        (SETQ VARIAVEIS (CONS (LIST 'VIAVEL 'NAO)
                (MAPCAN  '(LAMBDA(X)(CAR(CDR  X)))  VARIAVEIS)
)))
  (COND
    ((EQ  (CAR  RETORNO)  'SUCCESS)(SETQ   VARIAVEIS   (CONS
RETORNO VARIAVEIS)))
    ((AND (EQ (CAR RETORNO) 'FAIL)
          (NOT(MEMBERP    RESTR-CORRENTE     (GET      'RIGIDO
'BINDINGS))))
          (SETQ STATCEL (LIST 'STATUS 'NAO-VIAVEL))
          (SETQ VARIAVEIS (CONS RETORNO VARIAVEIS)) )) )

((EQ (SECOND(ASSOC 'VIAVEL VARIAVEIS)) 'NAO) VARIAVEIS)
```

```
((EQ (SECOND(ASSOC 'VIAVEL VARIAVEIS)) 'SIM) VARIAVEIS)
(SETQ CELA (FIND (CAR(SECOND E-IMG)) WB '(LAMBDA (X Y)(EQL
X (CAR Y)))))
(SETQ ROTA (RECUPERA-CAMINHO NCEL-IMG CELA))
(SETQ *HIGH-INTENSITY* T)
(MOSTRA_CELAS (MAPCAN '(LAMBDA (X)(FINDCELL X))ROTA))
(SETQ *HIGH-INTENSITY* NIL)
(SETQ RESP (EMITE-DIAG VARIAVEIS NCEL-IMG (SECOND(ASSOC
'INICIAL FRM-IMG))
            (SECOND(ASSOC 'OBJETIVO FRM-IMG))))
(MOSTRA_CELAS (MAPCAN '(LAMBDA (X)(FINDCELL X))ROTA))
((EQ RESP 'SIM)(CONS (LIST 'VIAVEL 'SIM) (LIST 'VIAVEL 'SIM)
                (MAPCAN '(LAMBDA(X)(CAR(CDR X))) VARIAVEIS))
)
(CONS (LIST 'VIAVEL 'NAO) (MAPCAN '(LAMBDA(X)(CAR(CDR X)))
VARIAVEIS)) )

(DEFUN EMITE-DIAG (VARIAVEIS-W NCEL-IMG C-INIC C-FIM
                   CONTLIN MSG N CELA RESP)
(SETQ CONTLIN 0 N 0)
(PUSH (LIST 'CELAS-DA-CONFIGURACAO (LENGTH LSTCELL))
VARIAVEIS-W)
(PUSH (LIST 'CELAS-ATINGIDAS (+ (LENGTH WB) (LENGTH WF)))
VARIAVEIS-W)
(PUSH (LIST 'CELAS-EXPANDIDAS (LENGTH WB)) VARIAVEIS-W)
(PUSH (LIST 'PONTO-INICIAL (CGC (SECOND(FINDCELL C-INIC))))
VARIAVEIS-W)
(PUSH (LIST 'PONTO-OBJETIVO (CGC (SECOND(FINDCELL C-FIM))))
VARIAVEIS-W)
(PUSH (LIST 'PONTO-AVALIADO (CGC (SECOND(FINDCELL
NCEL-IMG)))) VARIAVEIS-W)
(SETQ RATOM 'SIM)
(LOOP
(CLEAR-SCREEN)
(PRINT (NTH N VARIAVEIS-W))
(SETQ MSG (PROMPT-READ-CHAR '(P A J) "-PROXIMO  <A>-ANTERIOR
```

```
<J>-JULGA"))
(IF (EQ MSG 'A)
    (IF (> N 0) (DECQ N)))
(IF (EQ MSG 'J)
    (PROGN
      (CLEAR-SCREEN)
      (SETQ RESP (PROMPT-READ-CHAR '(V N) "<V>-VIAVEL
<N>-NAO VIAVEL"))))
(IF (EQ MSG 'P)
    (IF (< N (- (LENGTH VARIAVEIS-W) 2))
        (INCQ N)))
((EQ RESP 'V) 'SIM)
((EQ RESP 'N) 'NAO) ) )

(DEFUN LARGURA-MINIMA-ROTA (FRM-IMG E-VARS N-CEL E-CEL
                                                    RETOR
NO-W)
(SETQ RETORNO-W (LIST(LIST 'LARGURA-MINIMA-ROTA))
      LARGURA (SECOND(CAR(SECOND(ASSOC 'LARGURA-MINIMA-ROTA
FRM-IMG)))))
((OR (EQ (CAR E-CEL)(CAR(THIRD N-CEL)))(EQ (CAR
E-CEL)(THIRD(THIRD N-CEL))))
 (IF (<= LARGURA (- (CAR(SECOND(SECOND
N-CEL)))(CAR(FIRST(SECOND N-CEL))) ))
    (CONS 'SUCCESS RETORNO-W)
    (CONS 'FAIL RETORNO-W) ))
((OR (EQ (CAR E-CEL)(SECOND(THIRD N-CEL)))
                                        (EQ (CAR
E-CEL)(FOURTH(THIRD N-CEL))))
 (IF (<= LARGURA (- (SECOND(SECOND(SECOND N-CEL)))
     (SECOND(THIRD(SECOND N-CEL))) )) (CONS 'SUCCESS
RETORNO-W)
                                    (CONS 'FAIL RETORNO-W)
)) )

(DEFUN SEGMENTO-MINIMO (FRM-IMG E-VARS N-CEL E-CEL
                                SEGM CENTRO-G DIRECAO
NOVO-SEGM CENTRO-ANT-G
                                MINIMO OBJETIVO)
(SETQ E-VARS (SECOND(ASSOC 'SEGMENTO-MINIMO E-VARS))
      SEGM (SECOND(ASSOC 'SEGMENTO E-VARS))
      MINIMO (SECOND(CAR(SECOND(ASSOC 'SEGMENTO-MINIMO
FRM-IMG))))
      OBJETIVO (SECOND(ASSOC 'OBJETIVO FRM-IMG))
      CENTRO-G (CGC (SECOND N-CEL)))
(IF (NULL E-VARS)(SETQ CENTRO-ANT-G (CGC (SECOND E-CEL)))
    (SETQ CENTRO-ANT-G (SECOND(ASSOC 'CENTRO-ANT-G
E-VARS))))
(IF (EQ (CAR E-CEL) (FIRST (THIRD N-CEL))) (SETQ DIRECAO
'SUL)
    (IF (EQ (CAR E-CEL) (SECOND(THIRD N-CEL))) (SETQ DIRECAO
'OESTE)
        (IF (EQ (CAR E-CEL) (THIRD (THIRD N-CEL))) (SETQ
DIRECAO 'NORTE)
                                                (SETQ
DIRECAO 'LESTE))))
(SETQ NOVO-SEGM (IF (OR(EQ DIRECAO 'NORTE)
                        (EQ DIRECAO 'SUL))
```

```
                    (ABS(-      (SECOND      CENTRO-G)(SECOND
CENTRO-ANT-G)))
                    (ABS(-      (FIRST       CENTRO-G)(FIRST
CENTRO-ANT-G))))  )
((EQ (CAR N-CEL) OBJETIVO)
 (IF (AND(NULL(SECOND (ASSOC 'DIRECAO-ANTERIOR E-VARS)))(<
NOVO-SEGM MINIMO))
     (RETURN (LIST 'FAIL (LIST 'SEGMENTO-MINIMO)))
 (IF (EQ (SECOND(ASSOC 'DIRECAO-ANTERIOR E-VARS)) DIRECAO)
 (IF (< (+ SEGM NOVO-SEGM) MINIMO)
     (RETURN (LIST 'FAIL (LIST 'SEGMENTO-MINIMO)))
     (RETURN (LIST 'SUCCESS (LIST 'SEGMENTO-MINIMO))))
 (IF (OR (< SEGM MINIMO)(< NOVO-SEGM MINIMO))
     (RETURN (LIST 'FAIL (LIST 'SEGMENTO-MINIMO)))
     (RETURN (LIST 'SUCCESS (LIST 'SEGMENTO-MINIMO)))) ) ) )
((NULL (SECOND(ASSOC 'DIRECAO-ANTERIOR E-VARS)))
 (LIST  'SUCCESS  (LIST  'SEGMENTO-MINIMO  (LIST  (LIST
'DIRECAO-ANTERIOR DIRECAO)
 (LIST 'SEGMENTO  NOVO-SEGM)(LIST  'CENTRO-ANT-G  CENTRO-G))
)))
((EQ (SECOND(ASSOC 'DIRECAO-ANTERIOR E-VARS)) DIRECAO)
     (LIST  'SUCCESS  (LIST  'SEGMENTO-MINIMO  (LIST  (LIST
'DIRECAO-ANTERIOR
        DIRECAO) (LIST 'CENTRO-ANT-G CENTRO-G)
        (LIST 'SEGMENTO (+ (SECOND(ASSOC 'SEGMENTO  E-VARS))
NOVO-SEGM)))))  )
((AND(NEQ (SECOND(ASSOC 'DIRECAO-ANTERIOR E-VARS)) DIRECAO)
     (< (SECOND(ASSOC 'SEGMENTO E-VARS))
        (SECOND(CAR(SECOND(ASSOC          'SEGMENTO-MINIMO
FRM-IMG))))  ))
     (LIST  'FAIL  (LIST  'SEGMENTO-MINIMO  (LIST(LIST
'DIRECAO-ANTERIOR DIRECAO)
                  (LIST 'CENTRO-ANT-G CENTRO-G)
                  (LIST 'SEGMENTO NOVO-SEGM))))  )
(LIST  'SUCCESS  (LIST  'SEGMENTO-MINIMO  (LIST(LIST
'DIRECAO-ANTERIOR
      DIRECAO) (LIST 'CENTRO-ANT-G CENTRO-G)(LIST  'SEGMENTO
NOVO-SEGM))))  )

(DEFUN DISTANCIA-MINIMA-RECOMENDADA  (FRM-IMG  E-VARS  N-CEL
E-CEL
                                     MATCHING-LIST
REQ-CLAUSES FAIL-CLAUSES
                                     VIZINHOS     ROTULO-PAI
CENTRO-G DISTANCIA)
(SETQ               MATCHING-LIST                (SECOND(ASSOC
'DISTANCIA-MINIMA-RECOMENDADA FRM-IMG))
      CENTRO-G (CGC (SECOND N-CEL))
      VIZINHOS (THIRD N-CEL)
      ROTULO-PAI (SECOND(ASSOC 'PARTE-DE FRM-IMG)))
(LOOP
   ((NULL MATCHING-LIST)
     (COND
      ((NOT(NULL  FAIL-CLAUSES))(SETQ  RETORNO  (LIST  'FAIL
(LIST
                          'DISTANCIA-MINIMA-RECOMENDADA (LIST
                          (LIST 'DISTANCIA-MINIMA-RECOMENDADA
REQ-CLAUSES)
```

```
                              (LIST          'DISTANCIA-VERIFICADA
FAIL-CLAUSES)))) ))
       (T             (LIST             'SUCCESS             (LIST
'DISTANCIA-MINIMA-RECOMENDADA))) ))
   (SETQ  DISTANCIA  (DISTANCIA-CGC-CGO  N-CEL   (SECOND(CAR
MATCHING-LIST))
         CENTRO-G          (SECOND(CAR          (SECOND(ASSOC
'LARGURA-MINIMA-ROTA FRM-IMG))))))
   (IF (NOT(NULL DISTANCIA))
       (IF (< DISTANCIA (THIRD (CAR MATCHING-LIST)))
         (PROGN
           (PUSH (CAR MATCHING-LIST) REQ-CLAUSES)
           (PUSH    (LIST     (CAAR     MATCHING-LIST)(CADAR
MATCHING-LIST) DISTANCIA)
            FAIL-CLAUSES) ) ) )
   (POP MATCHING-LIST) ) )

(DEFUN DISTANCIA-CGC-CGO (CELA OBSTACULO CENTRO-G LARGURA
                                                    COORD  VIZ
ROTL)
(SETQ OBSTACULO (FIND OBSTACULO OBSTACLE
                      '(LAMBDA (X Y)(EQL X (PACK(CAR Y))))))
(SETQ ROTL (CAR OBSTACULO)
      COORD (SECOND CELA)
      VIZ   (THIRD CELA))
((EQUAL ROTL (CAR VIZ))
      (-(ABS(-(SECOND (FOURTH OBSTACULO))(SECOND CENTRO-G)))
LARGURA))
((EQUAL ROTL (SECOND VIZ))
      (-(ABS(-(CAR    CENTRO-G)(CAR(SECOND     OBSTACULO))))
LARGURA))
((EQUAL ROTL (THIRD VIZ))
      (-(ABS(- (SECOND(SECOND OBSTACULO))(SECOND CENTRO-G)))
LARGURA))
((EQUAL ROTL (FOURTH VIZ))
      (-(ABS(-(CAR     CENTRO-G)(CAR(THIRD     OBSTACULO))))
LARGURA)) )

(DEFUN DISTANCIA-MINIMA (FRM-IMG E-VARS N-CEL E-CEL
                         MATCHING-LIST VIZINHOS FAIL-CLAUSES
                         ROTULO-PAI    CENTRO-G     DISTANCIA
RETORNO MATCH-AUX-LIST)
(SETQ    MATCHING-LIST    (SECOND(ASSOC    'DISTANCIA-MINIMA
FRM-IMG)))
(SETQ CENTRO-G (CGC (SECOND N-CEL))
      MATCH-AUX-LIST MATCHING-LIST
      VIZINHOS (THIRD N-CEL)
      ROTULO-PAI (SECOND(ASSOC 'PARTE-DE FRM-IMG)) )
(LOOP
  ((NULL MATCHING-LIST)
     (COND
       ((NOT(NULL FAIL-CLAUSES))(SETQ RETORNO
         (LIST   'FAIL   (LIST   'DISTANCIA-MINIMA)))(RETURN
RETORNO))
       (T    (SETQ    RETORNO    (LIST    'SUCCESS     (LIST
'DISTANCIA-MINIMA)))) ))
  (SETQ  DISTANCIA  (DISTANCIA-CGC-CGO   N-CEL   (SECOND(CAR
MATCHING-LIST))
```

```
        CENTRO-G                          (SECOND(CAR(SECOND(ASSOC
'LARGURA-MINIMA-ROTA FRM-IMG))))))
  (DELETE  (UNPACK(SECOND  (CAR  MATCHING-LIST)))   VIZINHOS
'EQUAL)
  (IF (NOT(NULL DISTANCIA))
      (IF (< DISTANCIA (THIRD(CAR MATCHING-LIST)))
        (PROGN
         (SETQ MATCHING-LIST NIL)
         (SETQ FAIL-CLAUSES 'FAIL) ) ) )
  (POP MATCHING-LIST) )
(LOOP
  ((NOT (NULL FAIL-CLAUSES))RETORNO)
  (SETQ DEF-V (SECOND(ASSOC 'DEFAULT
                            (PARCIAL-MATCH-B      ROTULO-PAI
MATCH-AUX-LIST))))
  ((NULL DEF-V)(LIST 'SUCCESS (LIST 'DISTANCIA-MINIMA)))
  (LOOP
     ((NULL VIZINHOS)RETORNO)
     (COND
       ((EQ (CAAR VIZINHOS) 'O)
        (SETQ DISTANCIA  (DISTANCIA-CGC-CGO  N-CEL  (PACK(CAR
VIZINHOS)) CENTRO-G
                  (SECOND(CAR(SECOND(ASSOC
'LARGURA-MINIMA-ROTA FRM-IMG))))))
              (IF (AND(NOT(NULL DISTANCIA))
                  (NULL(MEMBERP (PACK(CAR VIZINHOS))
                                (PARCIAL-MATCH-B    ROTULO-PAI
MATCH-AUX-LIST))))
                  (IF (< DISTANCIA DEF-V)
                    (PROGN
                     (SETQ   RETORNO   (LIST   'FAIL    (LIST
'DISTANCIA-MINIMA)))
                     (SETQ VIZINHOS NIL
                           FAIL-CLAUSES 'FAIL) ) ) ) ) )
     (POP VIZINHOS) )((EQ (CAR RETORNO) 'SUCCESS)RETORNO)  )
)

(DEFUN AREA-PROIBIDA (FRM-IMG E-VARS N-CEL E-CEL
                                          MATCH-LIST)
(SETQ MATCH-LIST (GET 'AREA-PROIBIDA 'BINDINGS))
(LOOP
  ((NULL MATCH-LIST)(LIST 'SUCCESS (LIST 'AREA-PROIBIDA)))
  ((AND(EQ    (SECOND(ASSOC    'PARTE-DE    FRM-IMG))(CAAR
MATCH-LIST))
       (MEMBERP (CAR N-CEL) (SECOND(ASSOC (CADAR MATCH-LIST)
LST-AREAS))))
    (LIST 'FAIL (LIST 'AREA-PROIBIDA)))
  (POP MATCH-LIST) ) )

(DEFUN PARCIAL-MATCH (ROTULO BINDINGS
                                          LISTA)
(LOOP
  ((NULL BINDINGS)LISTA)
  (IF  (EQ  (CAAR  BINDINGS)  ROTULO)(PUSH  (CAR  BINDINGS)
LISTA))
  (POP BINDINGS) ) )

(DEFUN PARCIAL-MATCH-B (ROTULO BINDINGS
```

```
                                                    LISTA)
(LOOP
  ((NULL BINDINGS)LISTA)
  (IF (EQ (CAAR BINDINGS) ROTULO)(PUSH  (CDR(CAR  BINDINGS))
LISTA))
  (POP BINDINGS) ) )

(DEFUN  VIZINHANCA-NAO-RECOMENDADA  (FRM-IMG  E-VARS   N-CEL
E-CEL
                                          MATCH-LIST            VIZ
FAIL-CLAUSES STATCEL)
(SETQ MATCH-LIST  (SECOND(ASSOC  'VIZINHANCA-NAO-RECOMENDADA
FRM-IMG))
      STATCEL 'SUCCESS
      VIZ (MAPCAN '(LAMBDA (X)((NOT(ATOM X))(PACK X)))
                  (THIRD N-CEL)))
(LOOP
   ((NULL           VIZ)(LIST          STATCEL           (LIST
'VIZINHANCA-NAO-RECOMENDADA FAIL-CLAUSES)))
   (IF  (NOT(NULL  (MEMBER  (LIST  (SECOND(ASSOC   'PARTE-DE
FRM-IMG))(CAR VIZ))
                    MATCH-LIST 'EQUAL)))
      (PROGN
         (PUSH  (LIST  'VIZINHANCA-DETECTADA   (SECOND(ASSOC
'PARTE-DE FRM-IMG))
               (CAR VIZ)) FAIL-CLAUSES)
         (SETQ STATCEL 'FAIL)))
   (POP VIZ) ))

(DEFUN VIZINHANCA-PROIBIDA (FRM-IMG E-VARS N-CEL E-CEL
                            MATCH-LIST VIZ)
(SETQ    MATCH-LIST    (SECOND(ASSOC    'VIZINHANCA-PROIBIDA
FRM-IMG))
      VIZ (MAPCAN '(LAMBDA (X)((NOT(ATOM X))(PACK X)))(THIRD
N-CEL)))
(LOOP
  ((NULL VIZ)(LIST 'SUCCESS (LIST 'VIZINHACA-PROIBIDA)))
  ((NOT(NULL   (MEMBER   (LIST    (SECOND(ASSOC    'PARTE-DE
FRM-IMG))(CAR VIZ))
             MATCH-LIST 'EQUAL)))
      (LIST 'FAIL (LIST 'VIZINHANCA-PROIBIDA)))
  (POP VIZ) ) )

(DEFUN DIRECAO-PROIBIDA (FRM-IMG E-VARS N-CEL E-CEL
                                    P-DIRE DIRECOES FAIL-LIST)
(SETQ  P-DIRE   (SECOND(CAR(SECOND(ASSOC   'DIRECAO-PROIBIDA
FRM-IMG)))))
      DIRECOES '(SUL OESTE NORTE LESTE)
      VIZ (THIRD N-CEL))
(LOOP
   ((NULL DIRECOES)(LIST 'SUCCESS (LIST 'DIRECAO-PROIBIDA)))
   ((AND(EQ  P-DIRE  (CAR  DIRECOES))  (EQ   (CAR   VIZ)(CAR
E-CEL)))
   (LIST 'FAIL (LIST 'DIRECAO-PROIBIDA)))
        (POP DIRECOES)
        (POP VIZ) ) )

(DEFUN CALCULA-CUSTOS (CELA PAI FRM
```

```
                                FUNCAO-AVAL   CUSTOS   FUNCAO    PAIB
CUSTO-PAI)
;          ESTRUTURA            DA            CELA          :
(ROTULO-DA-CELA(ROTULO-DO-PAI((CUSTO1)(CUSTO2)...))
;
CUSTO-TOTAL))
; EX: (2 (1 ((1 3)(2 0)) 6))
(SETQ FUNCAO-AVAL (SECOND(ASSOC  'FUNCAO-GLOBAL-DE-AVALIACAO
FRM)))
((EQ PAI NIL)(LIST (CAR CELA)(LIST  PAI  (MAKE-LIST  (LENGTH
FUNCAO-AVAL)
             '(0 0)) 0)))
(SETQ PAIB (FINDCELL (CAR PAI))
      CUSTO-PAI (SECOND(SECOND PAI)))
(LOOP
   ((NULL FUNCAO-AVAL)(LIST (CAR CELA)(CONS (CAR PAI)
        (ADD-TOTAL-CUSTOS CUSTOS (SECOND(SECOND PAI))  ))))
   (SETQ FUNCAO (POP FUNCAO-AVAL))
   (SETQ CUSTOS (APPEND CUSTOS (LIST (LIST (* (CAAR FUNCAO)
    (EVAL (LIST (SECOND(CAR FUNCAO)) '(CAR CUSTO-PAI)  'CELA
'PAIB 'FRM)))
          (IF (NULL(SECOND FUNCAO)) 0 (* (CAADR FUNCAO)
              (EVAL  (LIST  (SECOND(SECOND  FUNCAO))  '(CAR
CUSTO-PAI)
                      'CELA 'PAIB 'FRM))))) )))
   (POP CUSTO-PAI) ) )

(DEFUN ADD-TOTAL-CUSTOS (CUSTOS CUSTOS-PAI
                                                TOTAL      TOTAL-P
N-CUSTO)
; TOT-ANT : CUSTO TOTAL DA CELA PAI
(SETQ TOTAL 0)
(LOOP
   ((NULL CUSTOS)(LIST N-CUSTO TOTAL))
   (SETQ   TOTAL-P    (LIST    (LIST    (+    (FIRST    (CAR
CUSTOS))(FIRST(CAR CUSTOS-PAI)))
                          (SECOND(CAR CUSTOS)))) )
   (SETQ TOTAL (+(+ (CAR(CAR TOTAL-P))(SECOND(CAR TOTAL-P)))
TOTAL))
   (SETQ N-CUSTO (APPEND N-CUSTO TOTAL-P))
   (POP CUSTOS)
   (POP CUSTOS-PAI) ) )

(DEFUN DISTANCIA-MANHATTAM (CUSTO-P FILHO PAI-B FRM-W)
; FRM-W -> FRAME DA SUB-ROTA
; PAI-B  -> ROTULO E ATRIBUTOS EM LSTCELL DA CELA PAI (E)
; FILHO -> ROTULO E ATRIBUTOS EM LSTCELL DA CELA FILHA (N)
; CUSTO-P -> LISTA DE  CUSTOS  OBTIDOS  PELO  COMPONENTE  DA
FUNCAO DE AVALIACAO
;             { ESTRUTURA (9 9) } DA CELA PAI
; DEVOLVE -> VALOR QUE REPRESENTA O CUSTO DE ATINGIR  N  VIA
E.
(DISTANCIA-CGC-CGC (SECOND FILHO)(SECOND PAI-B)) )

(DEFUN ESTIMATIVA-DISTANCIA-MANHATTAM (CUSTO-P  FILHO  PAI-B
FRM-W
                                       GOAL  COORD-O   CGC-C
CGC-O)
```

```
(SETQ GOAL (FINDCELL (SECOND(ASSOC 'OBJETIVO FRM-W)))
      CGC-C (CGC (SECOND FILHO))
      CGC-O (CGC (SECOND GOAL)))
(+(ABS (- (SECOND CGC-O)(SECOND CGC-C)))(ABS(- (CAR CGC-O)
  (CAR CGC-C)))) )

(DEFUN DISTANCIA-CGC-CGC (COORD-F COORD-P
                                              CGC-F CGC-P)
(SETQ CGC-F (CGC COORD-F)
      CGC-P (CGC COORD-P))
((EQL (CAR CGC-F)(CAR CGC-P))(ABS (- (SECOND CGC-F)(SECOND
CGC-P))))
((EQL (SECOND CGC-F)(SECOND CGC-P))(ABS (-(CAR CGC-F)(CAR
CGC-P)))) )

(DEFUN CGC (COORD)
(LIST (/ (+ (CAR(FIRST COORD))(CAR (SECOND COORD))) 2)
      (/ (+ (SECOND(SECOND COORD))(SECOND(THIRD COORD))) 2)
) )

(DEFUN CRUZAMENTO-PROIBIDO (FRM-IMG E-VARS N-CEL E-CEL
                                                LST-FRMS)
(SETQ LST-FRMS (CONS FRAME LIST-FRAMES))
((NOT(NULL(MAPCAN '(LAMBDA (X)((MEMBERP (CAR
N-CEL)(SECOND(ASSOC 'SOLUCAO X)))
                                    X)) LST-FRMS) ))
            (LIST 'FAIL (LIST 'CRUZAMENTO-PROIBIDO NIL)) )
(LIST 'SUCCESS (LIST 'CRUZAMENTO-PROIBIDO NIL)) )

(DEFUN COINCIDENCIA-PROIBIDA (FRM-IMG E-VARS N-CEL E-CEL
                              LST-FRMS SOLUCAO FRM-W NUM
STATCEL)
(SETQ LST-FRMS (CONS FRAME LIST-FRAMES)
      STATCEL 'SUCCESS)
(LOOP
   ((NULL (NTH 0 LST-FRMS))(LIST STATCEL (LIST
'COINCIDENCIA-PROIBIDA)))
   (SETQ FRM-W (POP LST-FRMS)
         SOLUCAO (SECOND(ASSOC 'SOLUCAO FRM-W))
         NUM 1)
  (LOOP
   ((NULL (NTH NUM SOLUCAO)))
   ((OR (EQ (CAR N-CEL) (CAR(LAST SOLUCAO)))(EQ (CAR
N-CEL)(CAR SOLUCAO)))
         (SETQ LST-FRMS NIL)(SETQ STATCEL 'FAIL))
   ((OR (AND (EQ (CAR N-CEL) (NTH NUM SOLUCAO))
             (EQ (CAR E-CEL) (NTH (- NUM 1) SOLUCAO)))
        (AND (EQ (CAR N-CEL)(NTH NUM SOLUCAO))
             (EQ (CAR E-CEL) (NTH (+ NUM 1) SOLUCAO))))
        (SETQ LST-FRMS NIL)(SETQ STATCEL 'FAIL))
   (INCQ NUM) ) ) )

(DEFUN NUMERO-DE-CRUZAMENTOS (CUSTO-P FILHO PAI-B FRM-W
                                                    LST-FR
MS)
(SETQ LST-FRMS (CONS FRAME LIST-FRAMES))
(+ (CAR CUSTO-P)(IF (NOT(NULL(MAPCAN '(LAMBDA (X)((MEMBERP
(CAR N-CEL)
```

```
                        (SECOND(ASSOC  'SOLUCAO  X)))(LIST  X)))
LST-FRMS))) 1 0)))

(DEFUN AREAS-DE-PREFERENCIA (CUSTO-P FILHO PAI-B FRM-W
                                                          MATCH-L
IST)
((MEMBERP  (CAR  FILHO)(SECOND(ASSOC   'AREAS-DE-PREFERENCIA
FRM-W)))
          (RETURN 0))
(* 0.2 (DISTANCIA-CGC-CGC (SECOND FILHO) (SECOND PAI-B))) )

(DEFUN ESTIMATIVA-AREAS-DE-PREFERENCIA (CUSTO-P FILHO  PAI-B
FRM-W
                                              GOAL  COORD-0   CGC-C
CGC-0)
(SETQ GOAL (FINDCELL (SECOND(ASSOC 'OBJETIVO FRM-W)))
      CGC-C (CGC (SECOND FILHO))
      CGC-0 (CGC (SECOND GOAL)))
(* 0.2 (+(ABS (- (SECOND CGC-0)(SECOND  CGC-C)))(ABS(-  (CAR
CGC-0)
  (CAR CGC-C)))))  )

(DEFUN RECUPERA-CAMINHO (E-IMG PAI)
((EQ E-IMG NIL)NIL)
(SETQ CELA (FIND (CAR(SECOND E-IMG)) WB '(LAMBDA (X Y)(EQL X
(CAR Y))) ))
(CONS (CAR E-IMG)(RECUPERA-CAMINHO CELA (CAR(SECOND CELA))))
)

(DEFUN FINDCELL (LBL)
(FIND LBL LSTCELL '(LAMBDA (X Y)(EQL X (CAR Y)))) )

(DEFUN EMISSAO-DE-RESULTADOS (FRM
                                    STATISTIC N SOLUCAO  A1  B1
A2 B2
                                    CGC-A CGC-B)
(IF (NOT(S-OU-N-P "IMPRIME RESULTADOS NA  TELA?"))(SETQ  WRS
'PRINT))
(PUSH (LIST 'PONTO-INICIAL (CGC (SECOND
          (FINDCELL     (CAR(LAST(SECOND(ASSOC     'SOLUCAO
FRM))))))))) STATISTIC)
(PUSH (LIST 'PONTO-OBJETIVO (CGC (SECOND
          (FINDCELL (CAR(SECOND(ASSOC  'SOLUCAO  FRM))))))))
STATISTIC)
(PUSH   (LIST   'CELAS-DA-CONFIGURACAO   (LENGTH   LSTCELL))
STATISTIC)
(PUSH (LIST 'CELAS-ATINGIDAS (+ WF-G WB-G)) STATISTIC)
(PUSH (LIST 'CELAS-EXPANDIDAS WB-G) STATISTIC)
(PUSH (LIST 'NO-CELAS-DA-ROTA (LENGTH (SECOND(ASSOC 'SOLUCAO
FRM))))
            STATISTIC)
(SETQ SOLUCAO (SECOND(ASSOC 'SOLUCAO FRM)))
(SETQ  STATISTIC  (APPEND  (RECUPERA-CUSTOS   FRM   SOLUCAO)
STATISTIC))
(SETQ STATISTIC (APPEND (APPEND (LIST 'SOLUCAO)
      (MAPCAN   '(LAMBDA     (X)(CGC(SECOND(FINDCELL     X))))
SOLUCAO)) STATISTIC))
```

```
(SETQ *HIGH-INTENSITY* T)
(LOOP
   ((NULL (SECOND SOLUCAO))(SETQ *HIGH-INTENSITY* NIL))
   (SETQ CGC-A (CGC (SECOND(FINDCELL (CAR SOLUCAO))))
         CGC-B (CGC (SECOND(FINDCELL (SECOND SOLUCAO)))))
   (POP SOLUCAO)
   (SETQ CGC-A (LIST (ABC CGC-A)(ORDEN CGC-A))
         CGC-B (LIST (ABC CGC-B)(ORDEN CGC-B)))
   (DRAW (LINE (FLOOR(CAR CGC-A)) (FLOOR(SECOND CGC-A))
               (FLOOR(CAR CGC-B))(FLOOR(SECOND CGC-B))) )
(SETQ N 0)
(LOOP
  ((NULL STATISTIC)(SETQ WRS NIL)(IF (NEQ  (PROMPT-READ-CHAR
'(F 0)
                          "<0> - OUTRA ROTA  <F>  -  FINALIZA")
'0)
                         (PROGN
                          (ALPHA-MODE)
                          (MAKE-WINDOW 0 0 25 80)
                          (LOAD MENUP)) )
  (IF (NOT (NULL (CAR STATISTIC)))
      (PROGN
          (PRINT (POP STATISTIC))
          (INCQ N)))
  (IF (AND(NEQ WRS 'PRINT)(> N 2))
           (PROGN
              (SETQ N 0)
              (CONTINUA-PROMPT)
              (CLEAR-SCREEN)) ) ) )

(DEFUN RECUPERA-CUSTOS (FRM ROTA
                                    CUSTOS CUSTOT FUNC VAL-SOL-P
VAL-SOL-T
                                    LISTA-W)
(SETQ CUSTOS (SECOND(SECOND(ASSOC (CAR ROTA) WB)))
      CUSTOT (THIRD(SECOND (ASSOC (CAR ROTA) WB)))
      FUNC   (SECOND(ASSOC 'FUNCAO-GLOBAL-DE-AVALIACAO FRM))
      VAL-SOL-P 0
      VAL-SOL-T 0)
(LOOP
   ((NULL   FUNC)(APPEND   (NREVERSE   LISTA-W)   (LIST(LIST
'CUSTO-TOTAL CUSTOT)) ))
   (SETQ LISTA-W (APPEND LISTA-W (LIST(LIST  (SECOND(CAR(CAR
FUNC)))
                 (/ (CAR(CAR CUSTOS)) (CAR(CAR(CAR  FUNC))))
)) ))
   (POP CUSTOS)
   (POP FUNC)     ) )

(RDS)
```