9,5
(nove e cinco)

ADRIANA JACOTO
ANDRÉ DE BESSA SANTOS

IMPLEMENTAÇÃO EM VISUAL PROLOG DE UM ROTEADOR DE HELICÓPTEROS

Trabalho de Formatura
apresentado à Escola Politécnica
da Universidade de São Paulo

Prof. Orientador: José Reinaldo Silva

São Paulo

1999

SUMÁRIO

1 Introdução

O roteador de helicópteros é constituído de um sistema para elaborar a programação de vôos de helicópteros entre uma base localizada em terra e plataformas de petróleo localizadas no mar. Este tipo de transporte consiste em levar passageiros a partir do aeroporto no litoral, denominado de base, até os heliportos localizados nas plataformas no mar e recolher passageiros a partir destes e levá-los até a base. O objetivo é atender à demanda de passageiros no menor tempo de atendimento e com o menor custo, considerando a disponibilidade de helicópteros.

O sistema roteador é uma aplicação de uma metodologia envolvendo a análise de uma estrutura em Redes de Petri e Inteligência Artificial na solução de um problema de "planning". Nesta abordagem, a partir da análise da estrutura da Rede de Petri do modelo do domínio, pode-se obter critérios de escolha da próxima ação a ser efetuada no plano que está sendo elaborado e, deste modo, dirigir o processo de busca da solução. A arquitetura do sistema é a de um planejador semi-reativo.

Na primeira etapa do trabalho, foram estudados tópicos relativos ao problema de planejamento em geral e à aplicação de técnicas de Inteligência Artificial (IA) na resolução de problemas de planejamento, relcaionados à implementação do roteador de helicópteros. Estes estudos foram realizados tendo como base a tese de doutorado de Lúcio Mitio Shimada [1], que propõe uma estruturação do problema de planejamento baseada em Inteligência Artificial e Redes de Petri. A tese também apresenta uma aplicação da abordagem proposta para um roteador de helicópteros.

Paralelamente ao estudo teórico também foi realizado um treinamento na linguagem de programação Visual Prolog, a partir da documentação especializada [11], [12], [13] e [14].

Na segunda etapa, foi feita a análise do programa em Visual Prolog feito por Shimada, quando da implementação elementar de um roteador de helicópteros em sua tese de doutorado. A partir da arquitetura do software e dos diagramas obtidos a partir desta engenharia reversa, foram relizados testes funcionais para a implementação "AntigoOK" do roteador.

Por fim, após a depuração e atualização desta versão preliminar foi desenvolvida a implementação de modificações do software original, incluindo a definição de pesos para as requisições de viagens, e a redefinição dos bolsões de helipontos a partir da análise do problema corrente. Foi também desenvolvida uma interface gráfica para o roteador de helicópteros utilizando os recursos do Visual Prolog.

2 O problema de planejamento

A aplicação de Inteligência Artificial (IA) a problemas de planejamento remete à idéia que está em geral associada a seres inteligentes: formas de vida capazes de tomar decisões elaborando planos de ação de modo que sua interação com o meio ambiente permita que estes seres possam lidar com os problemas do mundo real de maneira eficaz. Esse comportamento "inteligente" se opõe a mecanismos do tipo estímulo-resposta, característicos de seres inferiores.

Similarmente, para a solução de problemas de planejamento, são necessários sistemas capazes de escolher, dentre inúmeras possibilidades, a ação a ser tomada a cada instante, visando atingir determinado objetivo. As ações efetuadas pelo sistema, por sua vez, influenciam o meio-ambiente de forma variada e dinâmica, estando também este em estado de mudança contínua.

Assim, para a obtenção de um sistema de planejamento, torna-se necessário estabelecer um modelo do mundo (meio-ambiente em que se deseja atuar) e um modelo de ação (formas através das quais é possível atuar sobre este meio-ambiente).

É razoável esperar que a eficácia do planejador seja tanto maior quanto mais fiel forem esses modelos à realidade. Entretanto, para o caso em que se deseja obter tal sistema visando sua implementação com o auxílio do computador, como para o sistema roteador de helicópteros, é importante salientar que da simplicidade e capacidade de síntese desses modelos dependerá a eficiência do sistema, já que para que o planejador seja capaz de tomar decisões num ambiente complexo e mutável é necessário que estas sejam obtidas num tempo de processamento compatível.

Além disso, é preciso que estes modelos forneçam uma descrição clara, que auxilie a tomada de decisões para o caso de existência de metas conflitantes.

O modelo de mundo pode ser aberto, ou seja, abranger todos os eventos que ocorrem no mundo, incluindo aqueles sobre os quais o planejador não tem controle, ou fechado, em que apenas são consideradas as características do mundo conhecidas pelo planejador. O modelo de mundo fechado é mais restrito, e também menos complexo do ponto de vista computacional. Neste tipo de modelo, se o planejador não conhece a interpretação de determinada cláusula lógica, esta é assumida como sendo falsa.

O modelo de ação corresponde à descrição de um evento efetuado pelo planejador. Quando um evento ocorre, causa a modificação do mundo, de um estado para outro, ou seja, modifica determinadas características do mundo. Desse modo, um plano consiste numa sequência de ações a serem tomadas pelo sistema de planejamento, de modo que o mundo passe por uma determinada sequênca de estados que o leve de um estado inicial conhecido até um estado final desejado.

No processo de planejamento, a decisão do planejador acerca de qual curso de ação tomar é escolhida a partir de um vasto repertório de possibilidades. Esta decisão, por sua vez pode influenciar os estados de mundo de maneiras diversas e complicadas. Tudo isto é realizado num mundo complexo e dinâmico em que o meio-ambiente está em mudança contínua. O problema de planejamento possui, portanto, natureza combinatória, o que torna inviável a aplicação de algoritmos exatos para a resolução da maioria dos casos.

Desse modo, pode-se dizer que o problema de "planning" apresenta uma natureza não-linear e, geralmente, os espaços das possíveis soluções cresce combinatoriamente. Por este motivo, o

problema de "planning" requer um tipo de raciocínio não monotônico para o processo de elaboração de soluções.

Uma das principais aplicações dos sistemas de planejamento são em Sistemas Integrados de Manufatura (SIM). Nesses sistemas é preciso realizar o planejamento da produção, ou seja, determinar metas agregadas de produção para um certo intervalo de tempo. Também é necessário estabelecer a sequenciação das operações de manufatura, considerando cada peça a ser produzida e seu roteiro de produção, e também a alocação de recursos a estas operações. Para automatizar as funções de planejamento e sequenciação é imprescindível um sistema de planejamento capaz de efetuar essas funções de maneira integrada e que consiga oferecer garantia de qualidade, ou seja, que não cometa erros que comprometam o processo produtivo.

3 Sistemas de planejamento baseados em IA

A aplicação de Inteligência Artificial (IA) na solução de problemas de planejamento vem fornecer uma alternativa aos sistemas analíticos (que fornecem soluções exatas), computacionalmente inviáveis. Esses sistemas são especialmente inapropriados para a aplicação em ambientes dinâmicos, já que em geral utilizam algoritmos baseados em modelagens rígidas e que requerem elevado tempo de processamento. Ou seja, qualquer modificação ou perturbação no estado do mundo requereria uma nova solução do problema desde o começo.

As técnicas de Inteligência Artificial visam a obtenção de sistemas mais flexíveis, capazes de determinar não necessariamente a solução ótima, mas uma solução boa e viável do ponto de vista computacional. Além disso, esses sistemas devem ser capazes de realizar um replanejamento de forma a adaptar o plano corrente sendo construído, não sendo necessário abandoná-lo e começar um novo planejamento.

Porém, a aplicação de métodos elementares de IA não é suficiente para a solução de problemas de planejamento, devido a sua complexidade. Abordagens anteriores utilizando busca em largura, busca em profundidade, ou até mesmo a teoria linear de IA (que assume que uma meta pode ser dividida em submetas, sendo a solução para estas independentes) mostraram que é necessário adotar uma estruturação auxiliar para se chegar à solução do problema.

Nesse sentido, alguns sistemas baseados em IA foram propostos, como por exemplo o NOAH, em 1975. Este sistema utilizava níveis hierárquicos de abstração, ou seja, considerava o problema de planejamento em diferentes níveis de detalhamento, solucionando-o para cada nível e chegando, por "backtracking" a níveis menores de abstração.

Sendo assim, pode-se afirmar que os sistemas de planejamento baseados em Inteligência Artificial (IA) utilizam bastante conhecimento implícito e imcompleto, o que torna o desenvolvimento de sistemas para resolução deste tipo de problema muito complexo. Para se obter um melhor desempenho computacional nestes sistemas, é necessário se adotar mecanismos auxiliares para estruturar o problema.

3.1 A estruturação baseada em IA e Redes de Petri

O trabalho desenvolvido por Shimada [1], propões uma estruturação do problema de planejamento baseada em IA e no formalismo das Redes de Petri (RdP). Essa proposta de estruturação está baseada na construção de um *modelo conceitual (mc)* do problema, baseado em Redes de Petri. A RdP promove a modelagem do problema como um Sistema de Eventos Discretos (SED) e pode capturar a característica principal de um sistema produtivo que é o seu comportamento dinâmico.

O *mc* é constituído por dois modelos, que representam, separadamente, os tipos de conhecimento envolvidos no problema de "planning":

- "APPLICATION MODEL": representa o conhecimento do domínio da aplicação
- "PLANNING SELF-MODEL": representa o meta-conhecimento envolvido no processo de "planning", ou seja, representa um plano para fazer planos

A representação do domínio do problema através da Rede de Petri do "application model" torna possível a descrição dos modelos de mundo e de ação de uma maneira dinâmica. Isso significa que todos os estados e planos possíveis estão contidos na representação do

"application model". A partir da análise desta estrutura, ou seja da estrutura da Rede de Petri que representa o domínio do problema pode-se estabelecer uma estratégia de resolução do problema, isto é, aprender um meta-conhecimento. Utilizamos outra rede de Petri para elaborar o "planning self-model" que representa este meta-conhecimento.

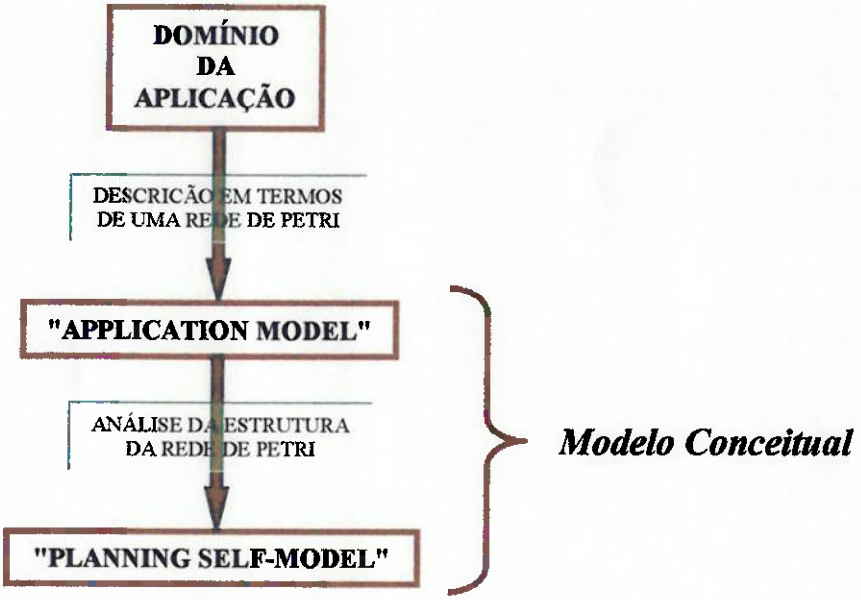


Figura 1 Modelo conceitual da estruturação baseada em IA e Redes de Petri

O *mc* é importante tanto para o processo de elaboração do plano inicial, assim como no caso ocorrer a necessidade de um replanejamento na falha, por exemplo, na execução de alguma tarefa do plano corrente.

A abordagem proposta apresenta diversas vantagens tais como: planos hierárquicos não lineares com diversos níveis de abstração, formalismo matemático, visualização gráfica, captura de todas as possíveis soluções, aprendizado de formas de decompor o problema e de estratégias de busca de soluções e capacidade de fazer o replanejamento na falha. Além disso, a abordagem apresenta a possibilidade de se implementar um planejador reativo puro e a possibilidade de se fazer a reutilização de planos.

Finalmente, a abordagem proposta não depende do domínio da aplicação, porque é a análise da estrutura das Redes de Petri, independentemente da sua interpretação para o domínio do problema, que permite o aprendizado do meta-conhecimento para dirigir o processo de planejamento. O planejado opera sobre a rede que representa o conhecimento do domínio, e não sobre a sua interpretação para um determinado domínio de aplicação.

4 O sistema STRIPS

O STRIPS – **STanford Research Institute Problem Solver** [2] foi o primeiro sistema de planejamento em IA bem sucedido. O objetivo do sistema é encontrar alguma composição de operadores que transformem um dado modelo de mundo inicial em outro que satisfaça alguma dada meta, expressa através de uma fórmula bem formada – fbf (ou *well-formed formula – wff*).

No STRIPS, um modelo de mundo é representado por um conjunto de fórmulas bem formadas do cálculo de predicados de primeira ordem. Um provador de teoremas, empregando o método de resolução, é utilizado para responder a questões dentro de um determinado modelo e a análise meios-fins empregada no GPS é usada para procurar através do espaço de modelos de mundo por aquele que satisfaça a meta. Assim os processos de prova de teoremas e procura através do espaço de modelos de mundo são inteiramente separados.

4.1 A operação do STRIPS

O espaço do problema para o STRIPS é definido por um modelo de mundo inicial, o conjunto de operadores disponíveis e seus efeitos nos modelos de mundo e a meta a ser atingida.

Como já mencionado, o STRIPS representa um modelo de mundo por um conjunto de fbfs. Por exemplo:

Fórmula bem formada (fbf)	Descrição
robo_esta_em (a)	Robô está na posição a
esta_em (B, b)	Objeto B está na posição b
esta_em (C, c)	Objeto C está na posição c

Tabela 1 Exemplos de fórmulas bem formadas utilizadas no STRIPS

Os operadores disponíveis podem ser agrupados em famílias chamadas *schemata*. Por exemplo o operador va_para (*m, n*), na verdade representa um conjunto de operadores diferindo entre si através dos pontos de saída (*m*) e/ou chegada (*n*). Diz-se que va_para (*m, n*) é um *schema* de operador cujos membros são obtidos substituindo-se constantes específicas pelos *parâmetros m* e *n*. No STRIPS, quando um operador for aplicado a um modelo de mundo, seus parâmetros já estarão instanciados com constantes específicas.

Cada operador é definido por uma descrição consistindo de duas partes principais: uma descrição dos efeitos do operador, e as sob as quais o operador é aplicável. Os efeitos de um operador são definidos simplesmente por uma lista de fbfs que devem ser adicionadas ao modelo e uma lista de fbfs que não são mais verdadeiras e portanto devem ser eliminadas. É conveniente enunciar a condição de aplicabilidade, ou *pré condição*, para um *schema* de operador como um *fbf shema*. Para determinar se há uma determinada instância de um operador que é aplicável num determinado modelo de mundo, deve-se provar que existe uma instância do *fbf schema* correspondente que pode ser provada a partir do modelo.

Como exemplo de um operador STRIPS, observe o operador ***empurre****(k,m,n)* que representa a ação do robô de empurrar um objeto *k* a partir de um lugar *m* até um novo lugar *n*, como mostrado a seguir:

Operador:	***empurre***(*k,m,n*)
Pré condição:	*robo_esta_em(m)*
	esta_em(k,m)
Lista de eliminação:	*robo_esta_em(m)*
	esta_em(k,m)
Lista de adição:	*robo_esta_em(n)*
	esta_em(k,n)

Da mesma forma, as metas são representadas por fbfs. Por exemplo, a tarefa: "Obter caixas *B* e *C* na posição *a*" poderia ser enunciada como a fbf:

$$esta\_em(B,a) \wedge esta\_em(C,a)$$

Resumindo, o espaço do problema para o STRIPS é definido por três entidades:

(1) Um modelo de mundo inicial, que consiste em um conjunto de fbfs descrevendo o estado presente do mundo.

(2) Um conjunto de operadores, incluindo uma descrição dos seus efeitos e dos seus fbf schemata de pré condição.

(3) Uma meta enunciada como fbf.

O problema está resolvido quando o STRIPS produz um modelo de mundo que satisfaz a meta em fbf.

4.1.1 A estratégia de busca

Para evitar a explosão combinatória resultante da aplicação de todos os operadores possíveis ao modelo de mundo inicial, gerando uma série de mundos sucessores aos quais também seriam aplicados todos os operadores possíveis, e assim por diante, até que a meta pudesse ser provada em algum modelo, o STRIPS adotou a estratégia GPS de extrair

"diferenças" entre o modelo de mundo atual e a meta e identificar operadores que sejam "relevantes" para reduzir essas diferenças. Uma vez que um operador relevante tenha sido determinado, tenta-se resolver o subproblema de produzir um modelo de mundo no qual ele seja aplicável. Se tal modelo for encontrado, aplica-se o operador relevante e reconsidera-se a meta original no modelo resultante.

O STRIPS começa empregando um provador de teoremas na tentativa de provar que a meta fbf G_0 deriva do conjunto M_0 de fbfs descrevendo o modelo de mundo inicial. Se G_0 realmente deriva de M_0 a tarefa está trivialmente resolvida no modelo inicial. Caso contrário, o provador de teoremas não conseguirá encontrar uma prova. Nesse caso, a prova incompleta é tomada como sendo a "diferença" entre M_0 e G_0. A seguir, operadores que podem ser relevantes para "reduzir" essa diferença são procurados. Estes são os operadores cujos efeitos no modelo de mundo permitiriam que a prova fosse continuada. Na determinação da relevância, os parâmetros dos operadores podem ser parcial ou totalmente instanciados. A correspondente pré condição instanciada em fbf schemata (do operador relevante) são então tomadas como sendo as novas submetas.

A hierarquia de meta, submetas e modelos gerados é representada por uma *árvore de busca*. Cada nó da árvore de busca tem a forma (<modelo de mundo>, <lista de metas>), e representa o problema de se tentar atingir as submetas na lista de metas (na ordem) a partir do modelo de mundo indicado.

Um exemplo de árvore de busca está representado na Figura 2. O nó do topo (M_0, (G_0)), representa a tarefa principal de atingir a meta G_0 a partir do modelo de mundo M_0.

Nessa árvore fica clara a geração de submetas (G_a, G_b, G_c, G_d, G_e, G_f) e a aplicação de operadores (OP_a, OP_b, OP_c, OP_e), gerando novos modelos de mundo (M_1, M_2, M_3, M_4), até que G_0 pôde ser provado em

M_4, gerando a solução do problema, que é consiste na aplicação de OP_c, OP_b, e OP_e, nessa ordem. Além disso, fica claro também que quando um operador é determinado como sendo relevante, ainda não é sabido onde ele ocorrerá no plano final, isto é, ele pode ser aplicável ao modelo inicial, e portanto ser o primeiro a ser aplicado, pode implicar na meta, e portanto ser o último a ser aplicado, ou pode estar em algum passo intermediário.

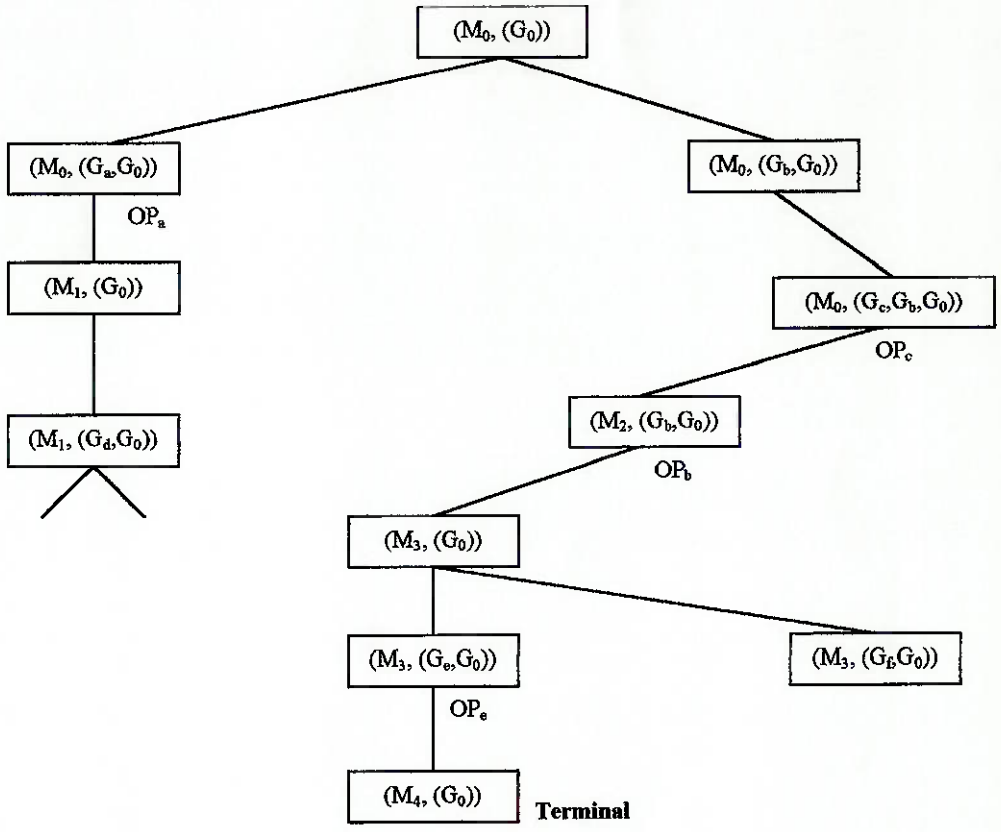


Figura 2 Uma típica árvore de busca STRIPS

Um fluxograma resumindo o processo de busca do STRIPS é mostrado na Figura 3.

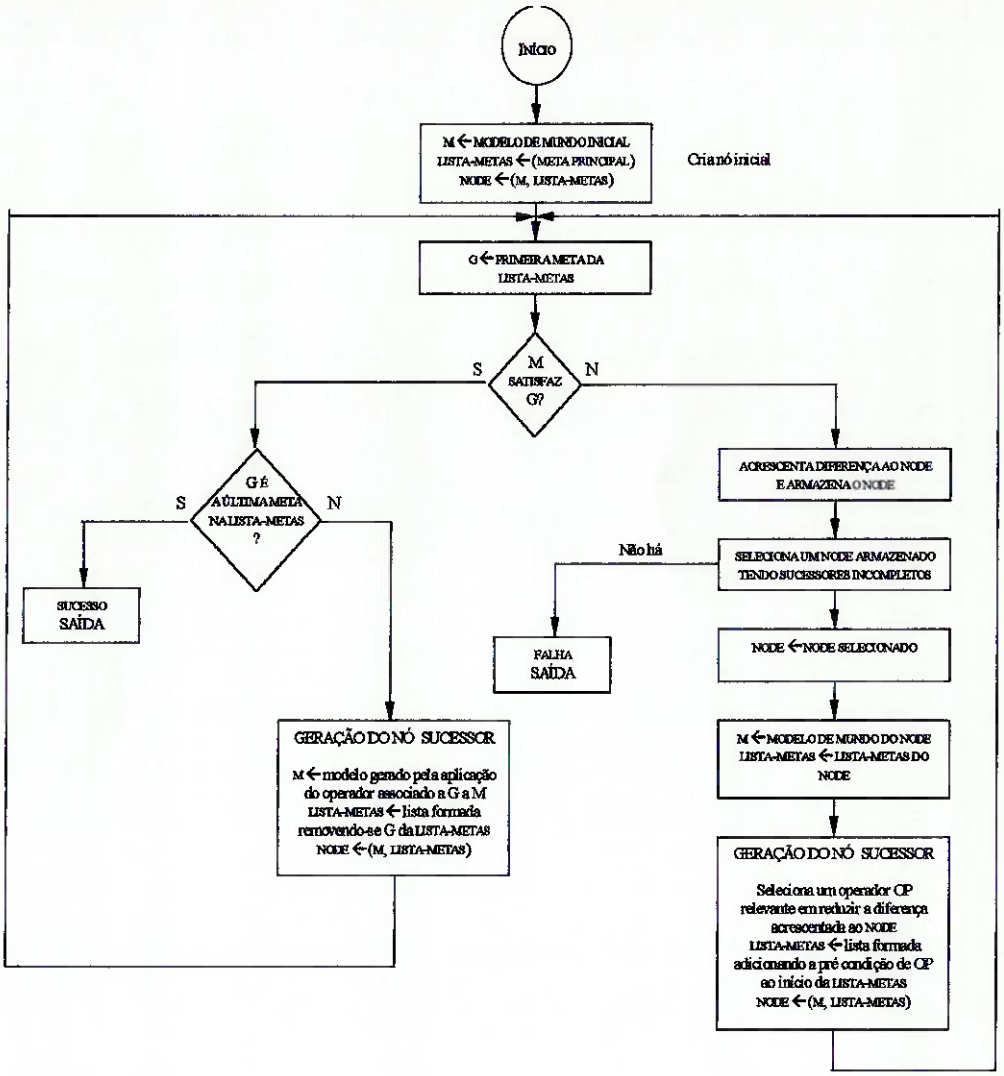


Figura 3 Fluxograma para o STRIPS

5 As Redes de Petri na modelagem de sistemas de planejamento

As Redes de Petri têm sido tradicionalmente utilizadas na modelagem de sistemas a eventos discretos. Alguns sistemas de planejamento baseado em IA têm também se utilizado das Redes de Petri para representar o plano elaborado, o modelo de mundo, ou ainda ações primitivas de operadores.

A grande vantagem do uso de Redes de Petri como ferramentas de modelagem em sistemas de planejamento reside em sua grande capacidade em representar o caráter dinâmico de um sistema. É possível, por exemplo, representar através de uma Rede de Petri todos os estados possíveis num modelo de mundo, utilizando a marcação da rede para determinar o estado corrente do sistema.

As Redes de Petri são especialmente adequadas também para modelagem em sistemas de planejamento reativos, já que esses são em geral elaborados para utilização em ambientes mutáveis, em que muitos eventos estão fora do controle do sistema controlador. O comportamento do meio ambiente pode ser modelado através da evolução das marcas em uma Rede de Petri representando o domínio do problema.

5.1 Tipos de Redes de Petri

As Redes de Petri Condição-Evento constituem o tipo mais elementar de Redes de Petri. Nelas, os estados do sistema são representados pelas condições, e a mudança de estados pelo disparo das transições. Para que determinada transição possa ser disparada, é necessário que ela esteja habilitada, o que significa que todas as suas pré-

condições devem ser verdadeiras e todas as suas pós-condições devem ser falsas. Após o disparo, todas as suas pré-condições cessam de ser verdadeiras e todas as suas pós-condições passam a ser verdadeiras, permanecendo inalteradas as demais condições.

Uma Rede de Petri Condição/Evento consiste então de um conjunto de condições, um conjunto de transições, um conjunto de arcos e uma marcação, podendo ser representada graficamente por um grafo.

A Rede de Petri Lugar-Transição é semelhante à Rede Condição-Evento, com a possibilidade de haver capacidade múltipla para as condições e pesos para os arcos. Ou seja, a marcação de uma Rede de Petri Lugar-Transição não é mais binária.

A Rede de Petri estendida orientada a objetos Ghenesys (General Hierarchical Extended Net System) é utilizada para a modelagem de sistemas mais complexos, para facilitar a análise estrutural de sistemas de grande porte. Este tipo de Rede de Petri possui duas classes de elementos: a classe box (box-capacidade ou box-de-tempo) e a classe atividade (atividade-simples ou atividade-composta). Possui ainda a possibilidade de análise em diversos níveis interligados de abstração.

5.2 A análise das Redes de Petri

A partir de um modelo de um sistema em Rede de Petri, é possível extrair diversas características comportamentais desse sistema a partir da análise dessa Rede de Petri. A seguir são apresentadas algumas características relevantes que podem ser obtidas:

- existência de conflito: duas transições são consideradas como estando em conflito quando, para uma determinada marcação, essas transições estão habilitadas e podem ser disparadas individualmente, mas não ao mesmo tempo

- invariantes de lugar: consiste numa relação algébrica que permanece constante, para determinados lugares na rede, qualquer que seja a marcação possível
- invariantes de transição: trata-se de uma sequência de eventos que pode ser repetida indefinidadmente, pois não modifica a marcação da RdP
- rede limitada: pode-se dizer que uma Rede de Petri é limitada se o número máximo de marcars em todos os seus lugares são limitados
- rede cíclica: consiste em uma rede na qual existe uma sequência de transições cíclica
- vivacidade: se refera à ausência de deadlocks na rede, pode-se dizer que uma rede é viva se todas as suas transições são livres de deadlocks
- reversibilidade: consiste na possibilidade de se alcançar novamente a marcação inicial da rede

6 A anomalia de Sussman

A maioria dos sistemas de planejamento utiliza uma representação do tipo STRIPS. Nesta representação, a descrição de uma ação apresenta: uma pré-condição, uma lista de adição e uma lista de eliminação. A lista de adição define cláusulas que poderiam não ser verdadeiras no modelo original mas que são verdadeiras no novo modelo que resulta após a aplicação do operador. A lista de eliminação especifica cláusulas do modelo original que não são mais verdadeiras no novo modelo.

Busca-se atingir o modelo do mundo desejado aplicando-se sucessivamente um conjunto de operadores STRIPS a partir do modelo do mundo inicial. Os sistemas aplicam ações para modificar o estado do mundo se suas pré-condições forem verdadeiras. O efeito da aplicação de uma ação está expresso nas listas de adição e de eliminação.

Entretanto, esta forma de representação apresenta problemas, sendo a sua principal deficiência o fato de não considerar a dependência entre metas concorrentes. Isto porque adotou-se a teoria linear que assume que quaisquer duas submetas de um problema podem sempre ser obtidas de uma forma independente. Desta forma, uma conjunção de metas era dividida em submetas para as quais se tentava obter uma solução isoladamente. É claro que esta hipótese não é verdadeira para a maioria dos problemas de "planning". Por este motivo, um sistema que emprega este tipo de representação não é capaz de contextualizar corretamente o efeito da aplicação de uma ação.

Uma consequência importante desta deficiência é a Anomalia de Sussman. Na Anomalia de Sussman, o atingimento de algumas metas pode requerer o estabelecimento de submetas para satisfazer pré-condições que irão desfazer cláusulas do estado meta anteriormente já

verdadeiras. Deste modo, o programa entra num processo de "looping" sem progresso em direção ao estado meta desejado.

Considerando-se o problema do Mundo de Blocos, suponha-se que existam três blocos na mesa, A, B e C, e que se deseja construir uma pilha onde A esteja sobre B, B sobre C, e este último sobre a mesa. Nenhuma conjunção é encontrada para resolver o problema. Então, é estabelecida uma estratégia de resolução. A teoria linear que determina as conjunções sugere que, primeiramente, mova-se A sobre B, e depois, B sobre C. Se as submetas são independentes, sua ordem não importa, então a ordem arbitrária é adotada. Desta forma, o robô primeiro coloca A sobre B. Em seguida, tenta colocar B sobre C, mas isto significa que ele necessita mover B. Mas o robô não pode mover B com A sobre ele (existe uma restrição física à mão do robô), e assim ele remove A de B e coloca A sobre a mesa. A seguir, coloca B sobre C e considera que a meta foi atingida. Entretanto, A não se encontra sobre B.

Este fenômeno acontece porque no modelo de ação do STRIPS, o efeito da aplicação de cada ação não está bem contextualizado. Ações distintas no seu efeito são erroneamente avaliadas como equivalentes. Todos os operadores apresentam igual oportunidade de serem escolhidos na hora de se decidir qual operador será aplicado em seguida. A abordagem meios-fins, criada no GPS, é o único critério empregado para se discriminar os operadores. Se o operador for considerado relevante para o atingimento do estado meta e se todas as suas pré-condições forem verdadeiras, então, o operador é aplicado logo em seguida.

7 Soluções existentes para o problema de planejamento

A seguir serão apresentados diversos sistemas de planejamento já apresentados na literatura. Estes podem ser divididos entre os sistemas clássicos, que abordam problemas simples usando técnicas como estratégia meios-fins, provadores de teorema e sistemas especialistas, e os sistemas integrados de planejamento e execução, que trabalham com planos hierárquicos parcialmente elaborados, sendo mais próximos dos sistemas reais cujo comportamento pretendem modelar.

7.1 Sistemas clássicos

- GPS: criado em 1963 por Newell e Simon, o General Problem Solver baseia-se na psicologia cognitiva, na pesquisa operacional e na lógica matemática. A principal contribuição deste sistema foi a abordagem meios-fins, que consiste numa forma de atingir uma meta estabelecendo submetas que, atingidas independentemente, levam ao sucesso da meta inicialmente proposta.
- STRIPS: criado em 1971 por Fikes e Nilsson, o STRIPS tem como princípio a representação do modelo de mundo e de ação a partir de fórmulas do cálculo de predicados, e a busca de operadores assim representados capazes de transformar um dado modelo do mundo inicial num modelo em que uma determinada meta possa ser provada como sendo verdadeira. Os operadores STRIPS são descritos em termos de um conjunto de pré-condições, uma lista de adição e uma lista de eliminação

7.2 Sistemas integrados de planejamento e execução

- NOAH: criado em 1975 por Sacerdoti, este sistema utiliza um grafo procedimental para fazer a representação do plano em diferentes níveis de abstração. Desse modo, o procedimento usado para se fazer o planejamento é hierárquico: inicia no nível mais alto de abstração e prossegue gerando planos cada vez mais detalhados para os demais níveis, de maneira "top-down". O sistema envolve ainda a crítica de planos, no que diz respeito à resolução de conflitos, ao uso de objetos existentes e a eliminar pré-condições redundantes.
- SIPE: criado em 1984 por Wilkins, é um sistema de planejamento independente do domínio que suporta a geração, automática ou interativa, de planos hierárquico parcialmente ordenados. O SIPE utiliza frames para fazer a especificação partcial de objetos e raciocinar acerca dos recursos.
- STRIPS+PLANEX: trata-se de um extensão ao STRIPS permitindo que o sistema fosse capaz de aprender com o processo de planejamento. Neste sistema, planos generalizados são armazenados numa forma específica chamada tabela triângulo. Um processo de especialização permite o emprego do plano generalizado.
- ART: sistema de planejamento baseado em restrições e aplicado para problemas de planejamento e sequenciação em transportes. Nessa abordagem, o domínio do problema é representado por um conjunto de variáveis e por um conjunto de condições que expressam os requisitos para os valores atribuidos a estas variáveis. A solução consite de um conjunto de atribuições para todas as variáveis do problema de tal maneira que todas as condições sejam satisfeitas

- PETRIX: sistema criado em 1988 por Rillo, para utilização nos níveis de controle de células de manufatura flexível, de estações de trabalho e de controle local. Utiliza Redes de Petri para a especificação das tarefas.
- PACEM: criado em 1993 por Paiva, o Planejador de Atividades de uma Célula de Montagem é um sistema de planejamento de atividades hierárquico e independente do domínio, que permite a elaboração de planos não lineares, parcialmente ordenados

8 O sistema híbrido IA / Redes de Petri

A análise da Rede de Petri que representa o domínio do problema pode sugerir estratégias de busca para dirigir o processo de solução do problema, por exemplo, para definir quais transições estão habilitadas em determinado instante.

A teoria das Redes de Petri é aplicada tradicionalmente ao problema de controle de sistemas discretos. A Rede de Petri permite a representação formal dos estados e da evolução de um sistema de eventos discretos. Permite explicitar relações entre estados, por exemplo: dependência de ordem parcial entre estados, existência de eventos concorrentes ou excludentes, compartilhamento de recursos, etc.

A análise estrutural de uma Rede de Petri pode prover informação importante para uma abordagem não-procedimental em sistemas de eventos discretos. O conhecimento sobre a estrutura do problema pode subsidiar formas de busca mais eficazes, e a minimização do "backtracking".

A Figura 4, a seguir, mostra a representação em Redes de Petri que constitui o modelo para o problema do Mundo de Blocos. A partir da figura, é possível observar-se que todos os estados podem ser atingidos a partir do estado em que todos os blocos estão sobre a mesa. Este estado é designado "nó essencial".

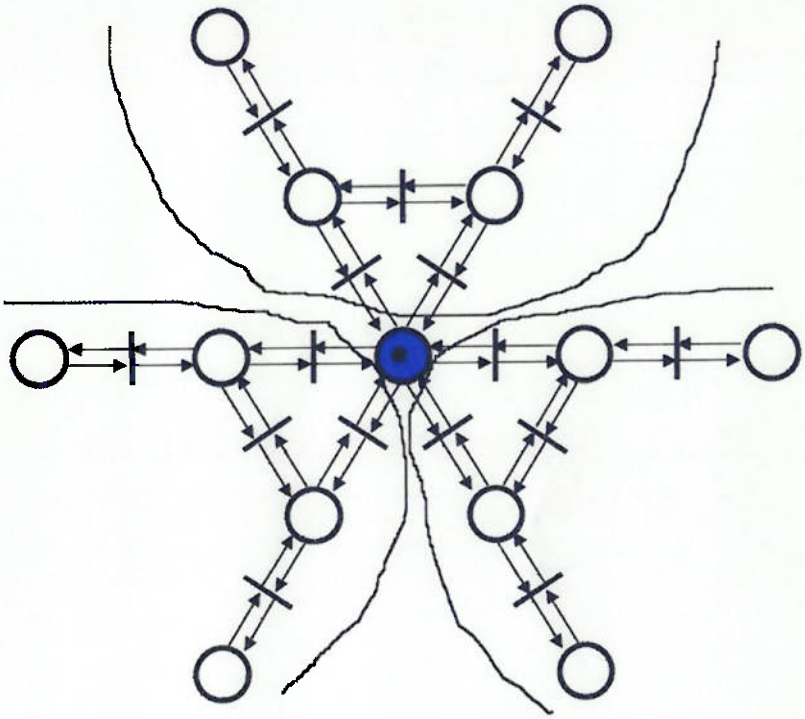


Figura 4 Rede de Petri para o Mundo de Blocos

Além disso, pode-se identificar três componentes conectadas ou subredes. Estas subredes definem agrupamentos de ações inter-relacionadas pela causalidade. Estas subredes estão interligadas através do "nó essencial". Desta forma, analisando a estrutura da Rede de Petri, podemos efetuar uma decomposição do problema em subproblemas que são independentes, de uma maneira tal que:

- estado onde todos os blocos estão sobre a mesa é um nó essencial do grafo
- primeiro bloco movido a partir do nó essencial determina uma sucessão de estados dependentes desta ação que estão na mesma componente conectada
- existe sempre um plano que corresponde à distância mínima entre dois estados que pertencem a uma mesma componente conectada

- plano envolvendo dois estados em duas componentes conectadas diferentes deve passar necessariamente pelo nó essencial

O conhecimento obtido a partir de uma análise da Rede de Petri que representa o domínio para este problema é muito importante. Como resultado da análise da Rede de Petri, pode-se sugerir o seguinte algoritmo para resolver este problema de planejamento:

Sejam dados EI = estado inicial, M = conjunção de metas e NE = nó essencial:

1. Se M já é verdadeiro (EI = M) então fim
2. Se EI e M estão na mesma componente conectada então existe um caminho entre EI e M sem passar pelo nó essencial. Resolva M usando um mecanismo do tipo meios-fins, obtendo o plano final P
3. Caso contrário, gere uma submeta NE = nó essencial obtendo um sub-plano Parcial-1. A seguir, resolva M a partir do nó essencial (NE) usando um mecanismo do tipo meios-fins obtendo um sub-plano Parcial-2. O plano final resulta da concatenação dos planos parciais P = Parcial-1 + Parcial-2

O algoritmo usa o conhecimento obtido a partir de uma análise da Rede de Petri que representa o conhecimento do domínio e é capaz de evitar o problema da Anomalia de Sussman. Este algoritmo faz a decomposição do problema baseado na análise da estrutura da rede. Este conhecimento está baseado na noção de transições independentes para assegurar a independência de cada uma das subredes e na existência do "nó essencial" que faz a conexão entre todas estas subredes. A interdependência pela causalidade ocorre apenas entre ações que pertencem à mesma componente conectada do grafo

9 Implementação elementar do roteador de helicópteros

O sistema Roteador de Helicópteros objeto deste trabalho foi originalmente desenvolvido como aplicação na Tese de Doutorado de Lúcio Mitio Shimada, apresentada à Escola Politécnica da Universidade de São Paulo em 1997. O Roteador de Helicópteros é um exemplo de uma aplicação da teoria de Inteligência Artificial a um problema de planejamento. A mesma metodologia baseada em IA e Redes de Petri aplicada ao problema do Roteador pode ser utilizada na solução de problemas de planejamento em automação, robótica e sistemas integrados de manufatura.

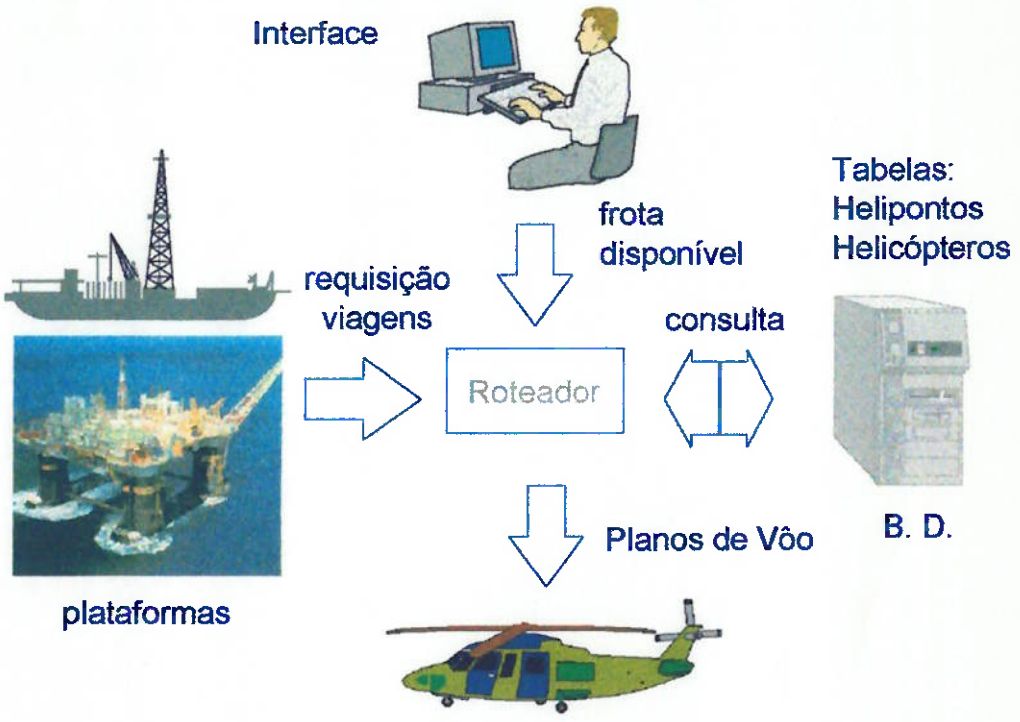


Figura 5 Roteamento de helicópteros na Bacia de Campos

Além de ser um exemplo prático de aplicação de uma metodologia teórica, o Roteador é também a solução de um problema

real de roteamento de helicópteros na Bacia de Campos (vide Figura 5). Este problema consiste em elaborar a programação de vôos dos helicópteros que fazem o transporte de passageiros entre a base localizada em terra e as plataformas de petróleo localizadas no mar.

Para se ter uma idéia da complexidade do problema, estão envolvidos 50 helipontos, 16 helicópteros de 7 tipos diferentes, 5 horários de partida e até 20 mil passageiros transportados por mês. O objetivo do roteador é, a partir da demanda de passageiros de ida para as plataformas e de volta para a base, fornecer as rotas dos helicópteros, visando o menor tempo de atendimento e o menor custo, considerando também a disponibilidade dos mesmos e a obtenção de uma solução boa e computacionalmente viável.

Sendo assim, são entradas do programa:

- dia e horário de partida do vôo
- dados dos helipontos nas plataformas
- dados da frota de helicópteros
- requisições de viagens

As saídas que o programa deve fornecer são:

- dados das rotas geradas (tipos dos helicópteros atribuídos, horário de partida de cada vôo, sequência de helipontos nas rotas, grade de horários para cada heliponto na rota, distância entre os helipontos, tempo de viagem em cada trecho da rota, carga dos helicópteros em cada trecho da rota)
- custo total (US$) das rotas geradas
- distância total percorrida
- peso transportado (kg)
- relação custo/peso (US$/kg)

De acordo com a metodologia utilizada, deve-se em primeiro lugar considerar o domínio do problema. A Figura 6 ilustra um modelo

da Bacia de Campos, em que os diversos helipontos estão agrupados em sub-redes, a partir de sua localização geográfica.

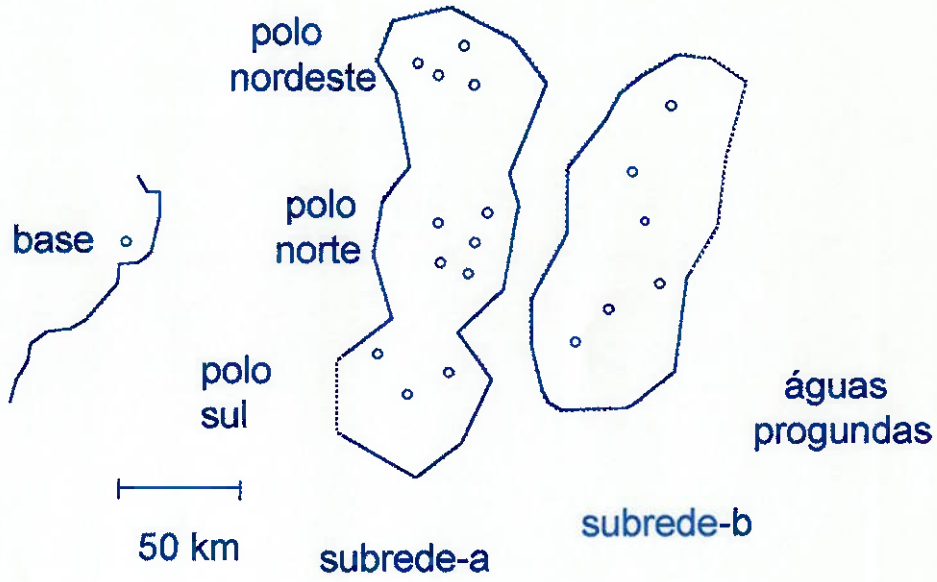


Figura 6 Domínio da aplicação

Em seguida, descrevendo-se este domínio da aplicação em termos de Redes de Petri, obtém-se o Application Model. A Figura 7 ilustra o Application Model para uma sub-rede.

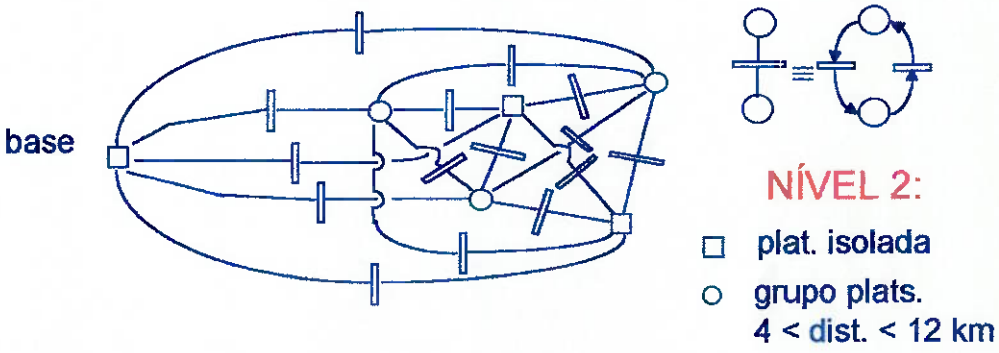


Figura 7 Application Model

Analisando-se o Application Model, observa-se que este é bastante semelhante ao obtido para o problema do Mundo de Blocos (vide Figura 4), sendo a base o "nó essencial" que interliga as sub-redes.

O Planning Self-Model pode ser representado pela Rede de Petri ilustrada na Figura 8.

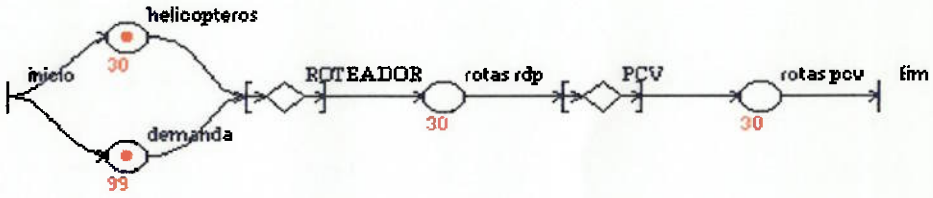


Figura 8 Componentes do roteador de helicópteros

O módulo Roteador da Rede de Petri da Figura 8 pode ser visto na Figura 9. Observa-se que a Rede de Petri que constrói os planos (Planning Self-Model da Figura 9) opera sobre a Rede de Petri do domínio (Application Model da Figura 7).

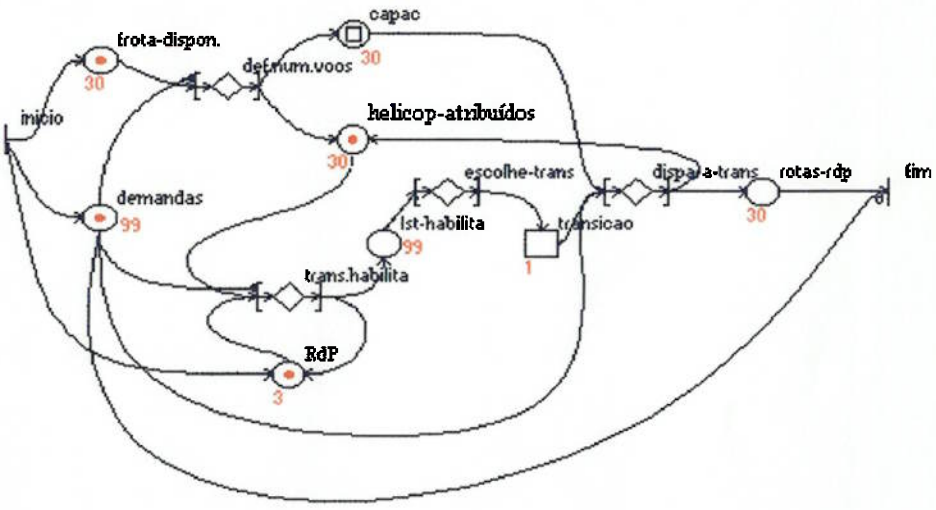


Figura 9 Rede de Petri do Roteador

Sendo assim, a Rede de Petri que representa o domínio (Application Model) é construída (na forma de fatos do Visual Prolog) cada vez que o rotemento é feito. A construção desta Rede depende de quais sub-redes estão envolvidas no problema corrente, ou seja, de quais helipontos contêm requisições de viagens. Isto pode ser visto na Figura 10.

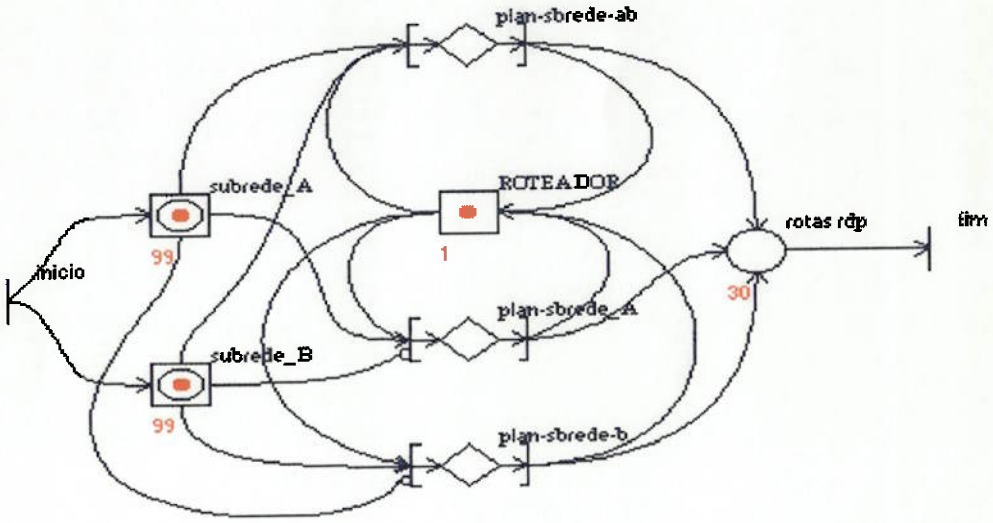


Figura 10 Tratamento das sub-redes

Além disso, a resolução do problema é feita para diversos níveis de abstração, como descrito na Figura 11. Para cada nível de abstração, foram definidos modelos do domínio correspondentes. Nos modelos de alto nível, os lugares são constituídos por grupos de helipontos com coordenadas geográficas de localização próximas.

Os níveis de abstração usados foram três: Nível 0, Nível 1 e Nível 2. No Nível 0 ou nível "ground" os boxes da Rede de Petri são constituídos pelas plataformas tratadas isoladamente (•). No Nível 1, os boxes representam plataformas isoladas (•) e grupos de plataformas distantes entre si até 4 km (•). No Nível 2, os boxes representam

plataformas isoladas (•), grupos de plataformas distantes entre si até 4 km (•) e grupos entre 4 km e 12 km (•).

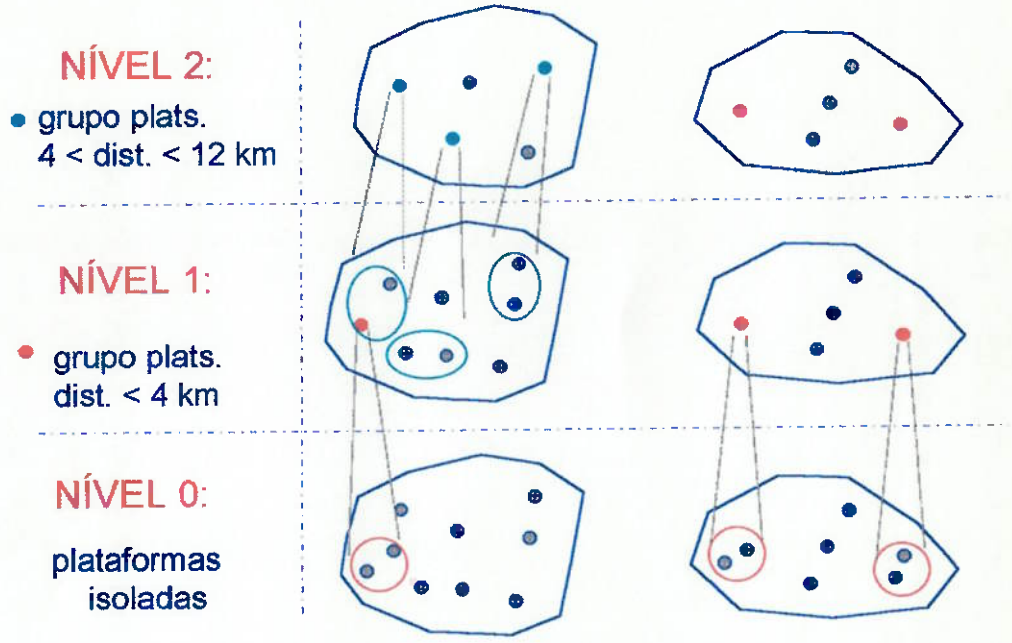


Figura 11 Definição de níveis de abstração

A Figura 12 mostra a Rede de Petri que inclui o processo de agrupamento de helipontos e de síntese do modelo do domínio.

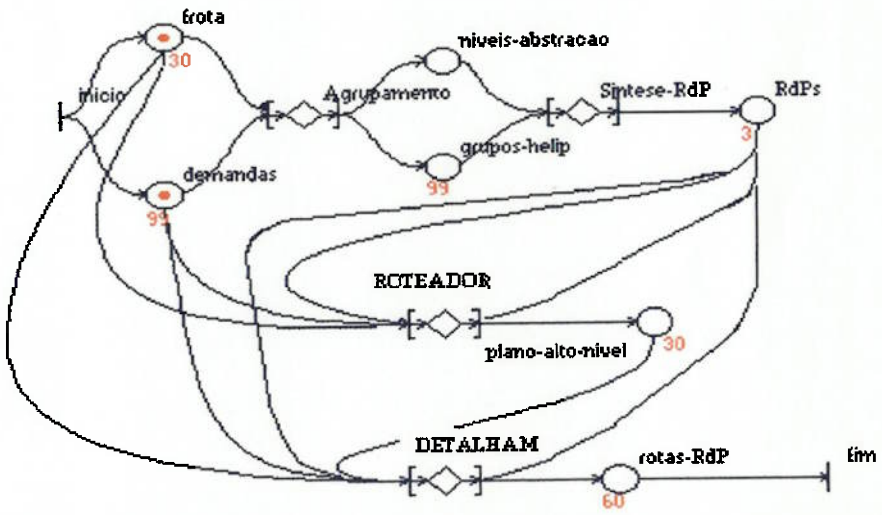


Figura 12 Estruturação do conhecimento do domínio

Por fim, após a obtenção das rotas a partir do módulo do Roteador, é realizada uma pós-otimização das mesmas (vide módulo PCV na Figura 8), com o auxílio do algoritmo das permutações para o PCV (Problema do Caixeiro Viajante), ilustrado na Figura 13.

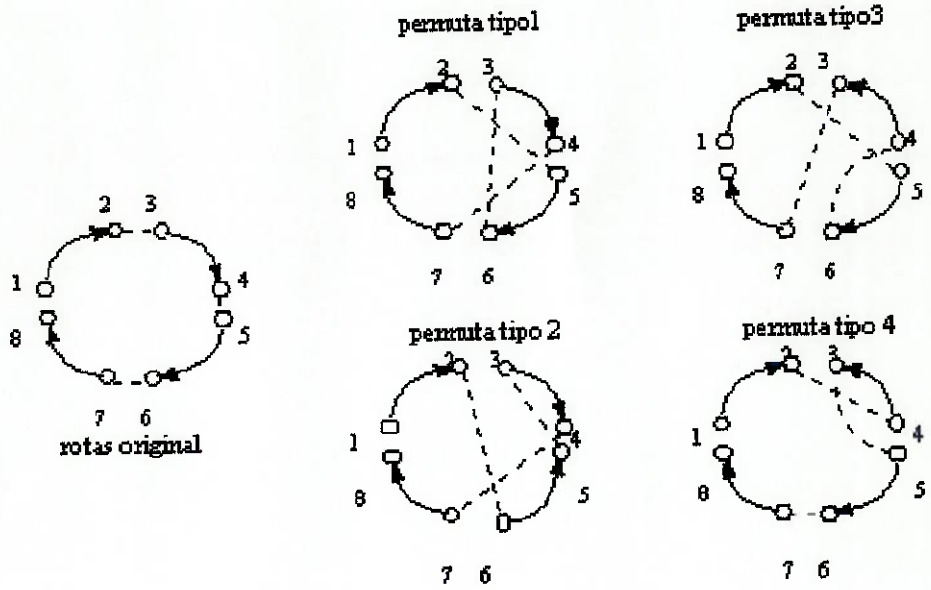


Figura 13 Pós-otimização das rotas

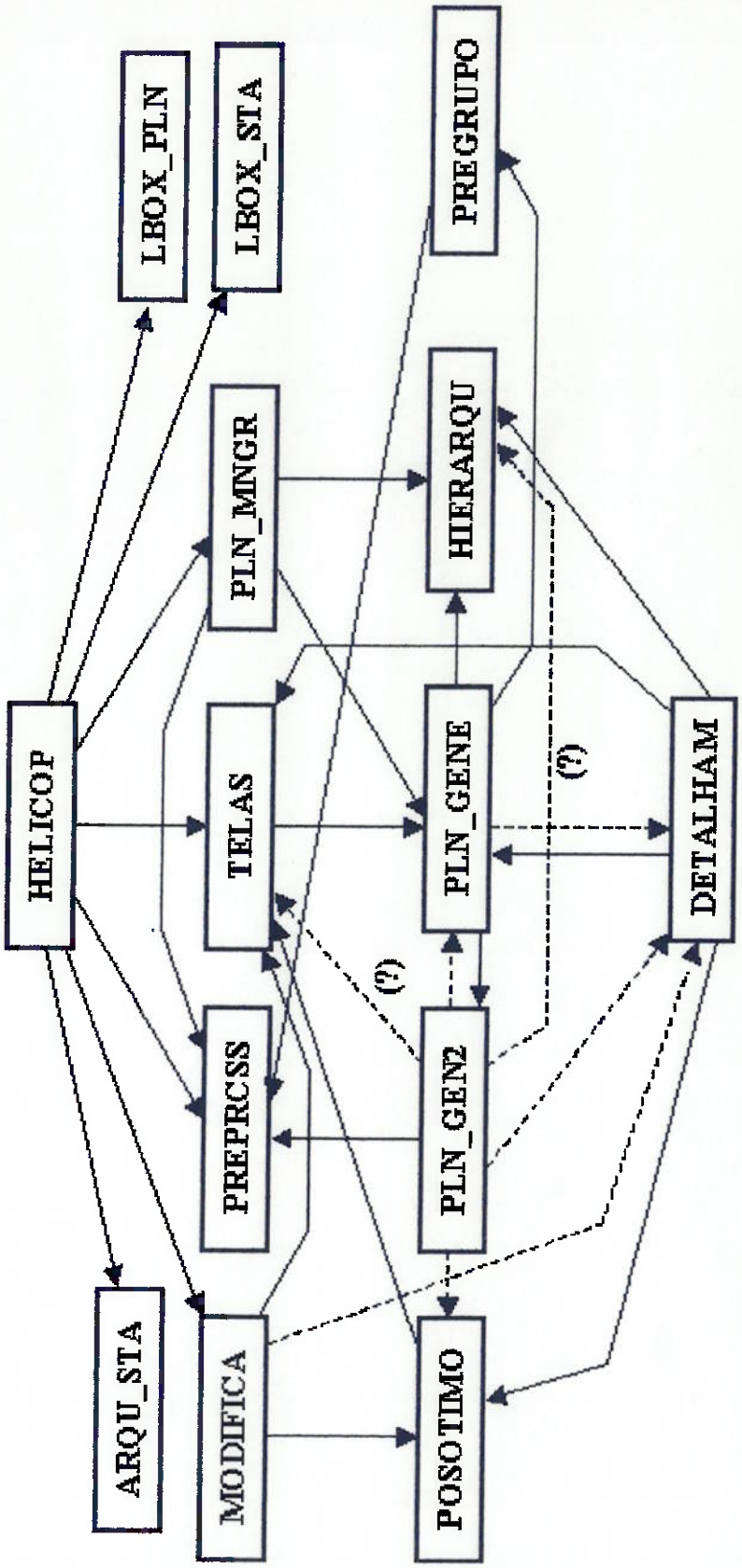
10 A arquitetura do programa em Visual Prolog

Os diagramas a seguir ilustram a arquitetura do programa em Visual Prolog da implemetação elementar "AntigoOK" feita na tese de doutorado de Lúcio Mitio Shimada [1] para o problema do planejador semi-reativo estruturado em Redes de Petri estendidas Ghenesys.

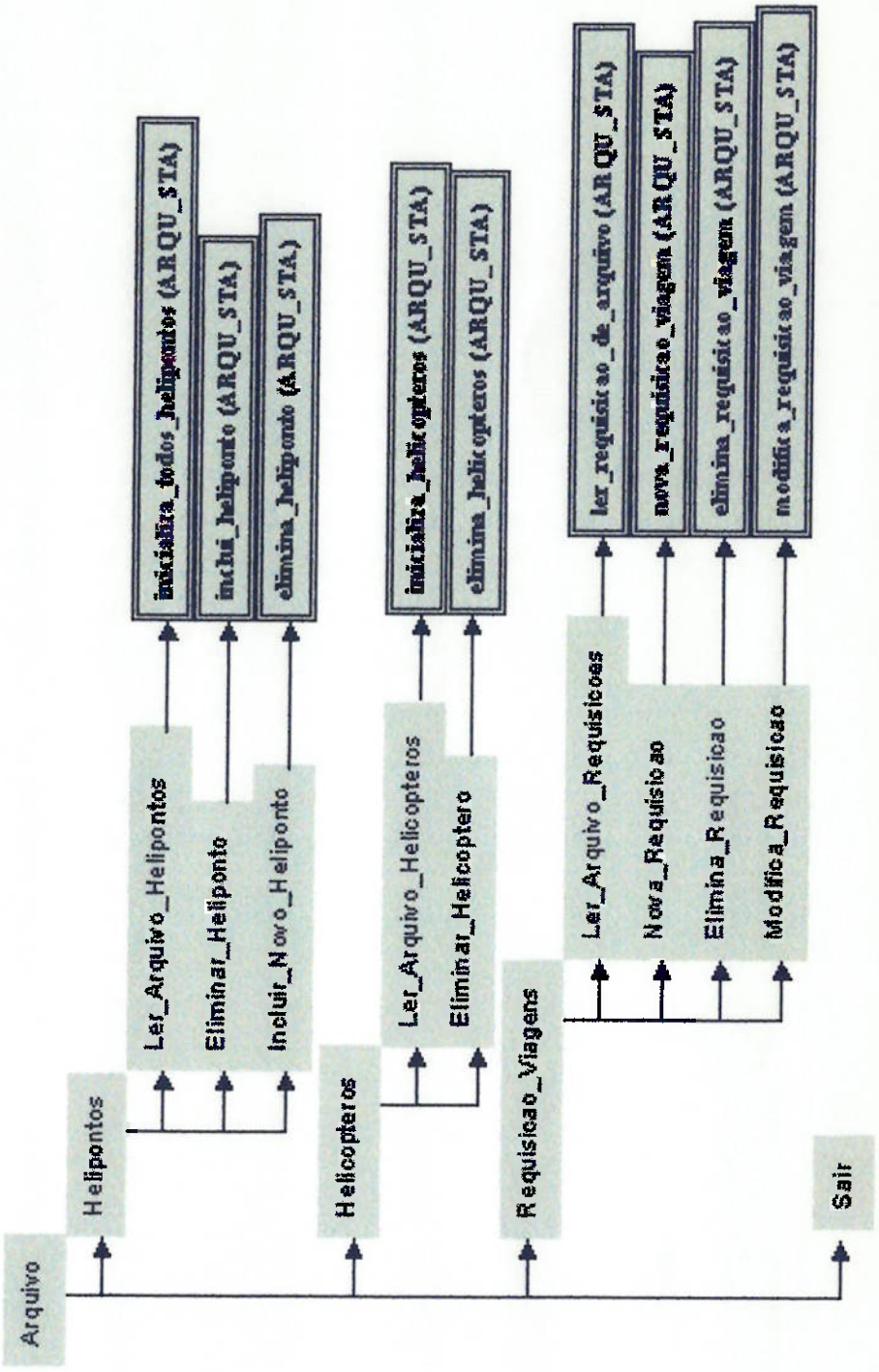
Legenda :

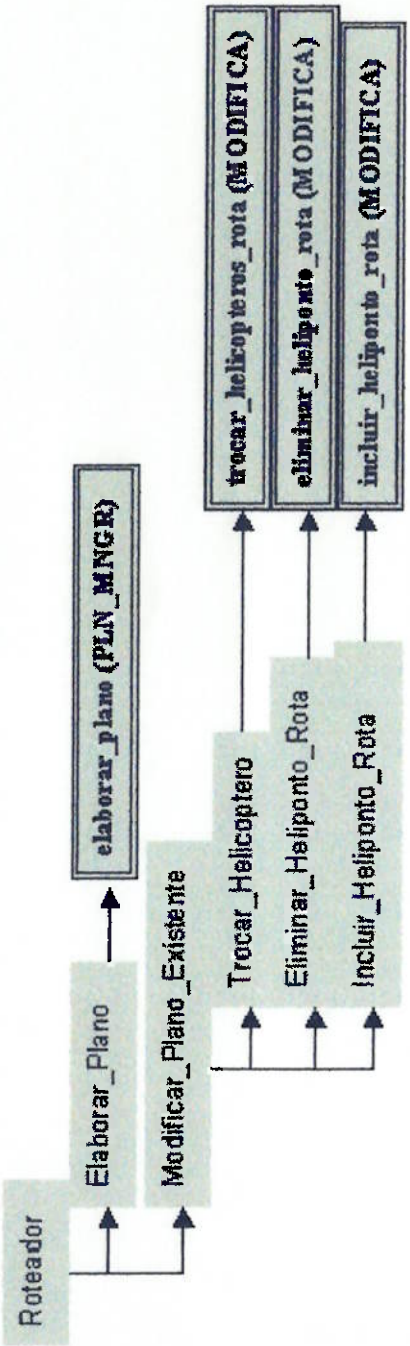
Exemplo	Definição
PLN_MNGR	Arquivo
Arquivo	Item de menu
fill_lbox_requisicao (LBOX_STA)	Predicado global declarado em outro arquivo
dlg_helicopteros_Create	Predicado global declarado no próprio arquivo
inicia_dialogo_capacidade	Predicado local
⟶	Chamada ativa
⇢	Chamada inativa (sob comentário)
HELICOP	Arquivo chamador (ativo)
(DETALHAM)	Arquivo chamador (inativo)

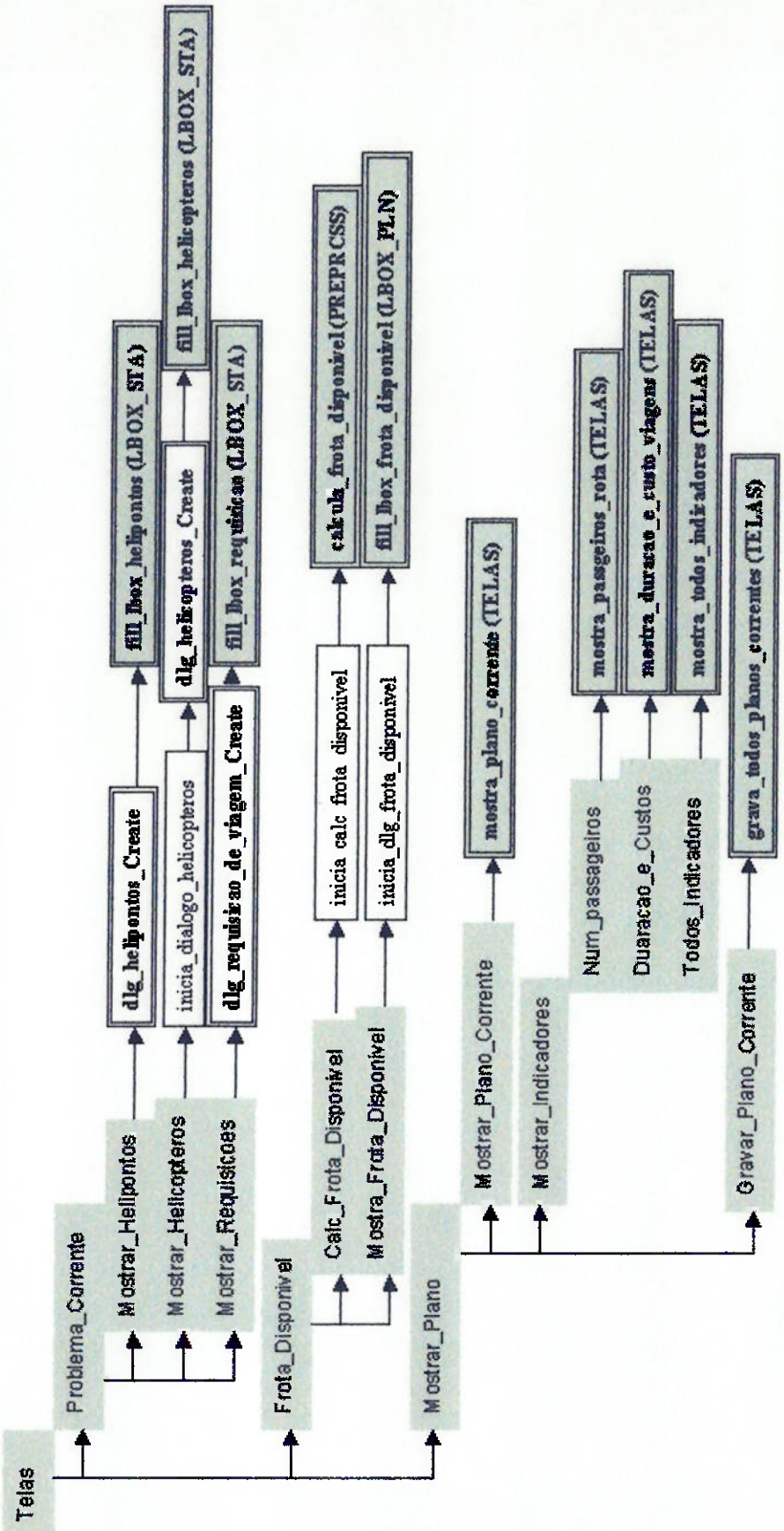
DIAGRAMA GERAL DOS PROGRAMAS

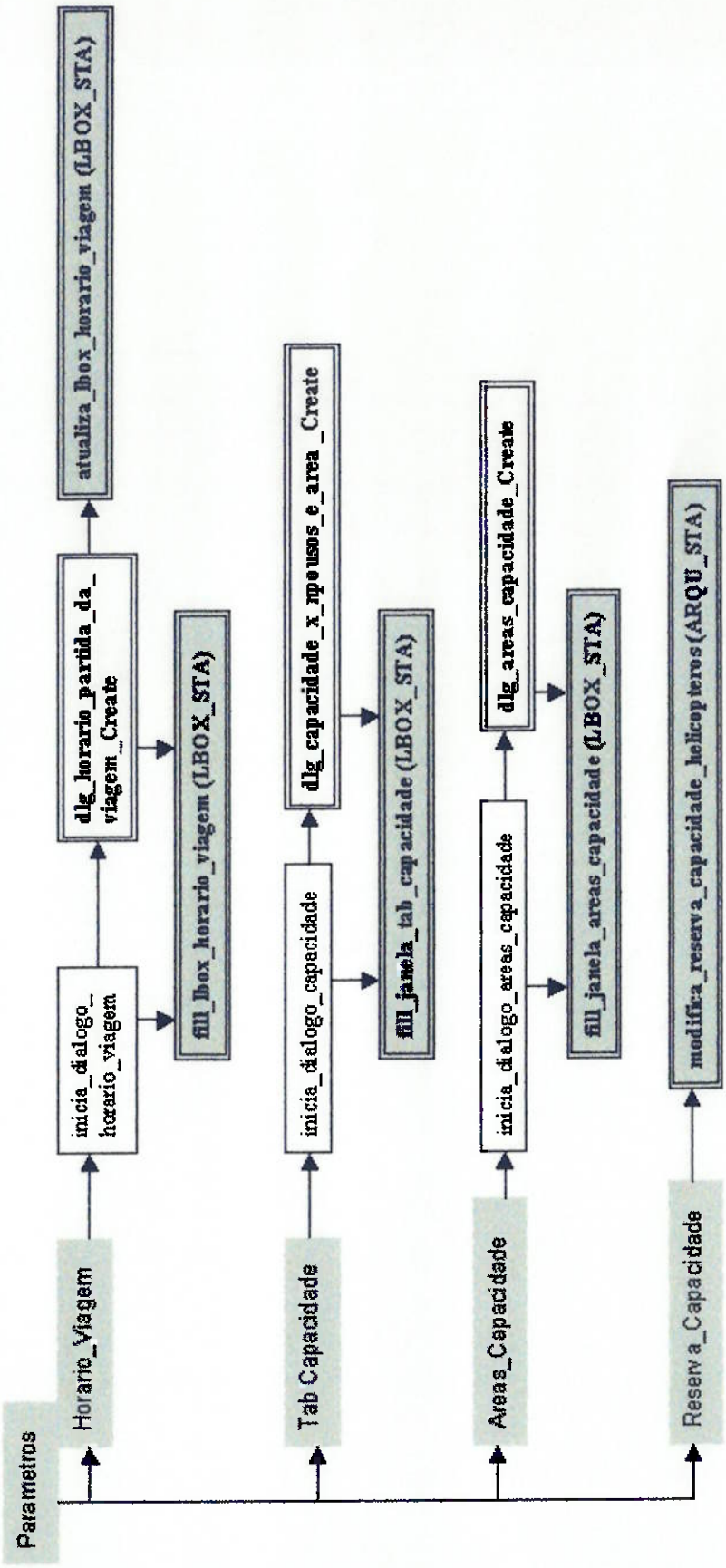


Arquivo: HELICOP.PRO (principal)



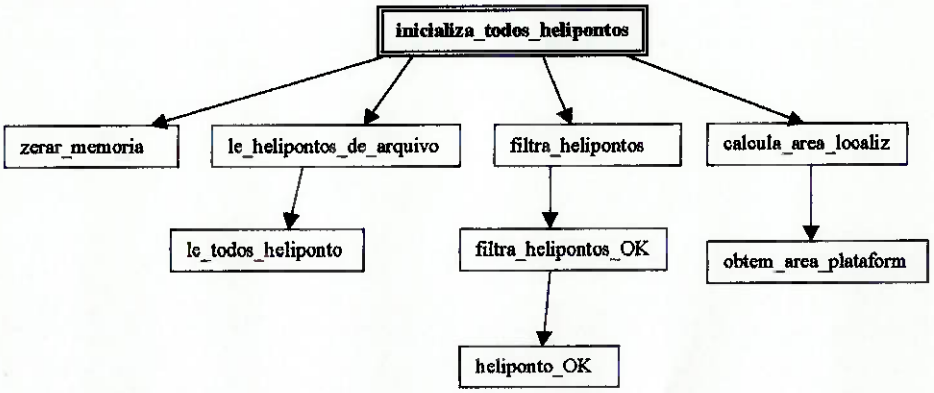

Roteador
Elaborar_Plano
elaborar_plano (PLN_MNGR)
Modificar_Plano_Existente
Trocar_Helicoptero
Eliminar_Heliponto_Rota
Incluir_Heliponto_Rota
trocar_helicopteros_rota (MODIFICA)
eliminar_heliponto_rota (MODIFICA)
incluir_heliponto_rota (MODIFICA)


Telas
Problema_Corrente
Mostrar_Helipontos
Mostrar_Helicopteros
Mostrar_Requisicoes
dlg_helipontos_Create
inicia_dialogo_helicopteros
dlg_requisicao_de_viagem_Create
fill_lbox_helipontos (LBOX_STA)
dlg_helicopteros_Create
fill_lbox_requisicao (LBOX_STA)
fill_lbox_helicopteros (LBOX_STA)
Frota_Disponivel
Calc_Frota_Disponivel
Mostra_Frota_Disponivel
inicia calc frota disponivel
inicia_dlg_frota_disponivel
calcula_frota_disponivel (PREPRCSS)
fill_lbox_frota_disponivel (LBOX_FLN)
Mostrar_Plano
Mostrar_Plano_Corrente
Mostrar_Indicadores
mostra_plano_corrente (TELAS)
Num_passageiros
Duaracao_e_Custos
Todos_Indicadores
mostra_passgeiros_rota (TELAS)
mostra_duracao_e_custo_viagem (TELAS)
mostra_todos_indicadores (TELAS)
Gravar_Plano_Corrente
grava_todos_planos_correntes (TELAS)


Parametros
Horario_Viagem
Tab Capacidade
Areas_Capacidade
Reserva_Capacidade
inicia_dialogo_horario_viagem
inicia_dialogo_capacidade
inicia_dialogo_areas_capacidade
dlg_horario_partida_da_viagem_Create
fill_lbox_horario_viagem (LBOX_STA)
atualiza_lbox_horario_viagem (LBOX_STA)
dlg_capacidade_x_npeuss_e_area_Create
fill_janela_tab_capacidade (LBOX_STA)
dlg_areas_capacidade_Create
fill_janela_areas_capacidade (LBOX_STA)
modifica_reserva_capacidade_helicopteros (ARQU_STA)

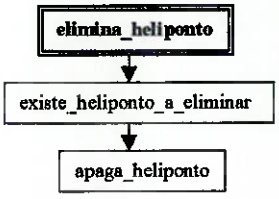
Arquivo: ARQU_STA.PRO

1) HELICOP

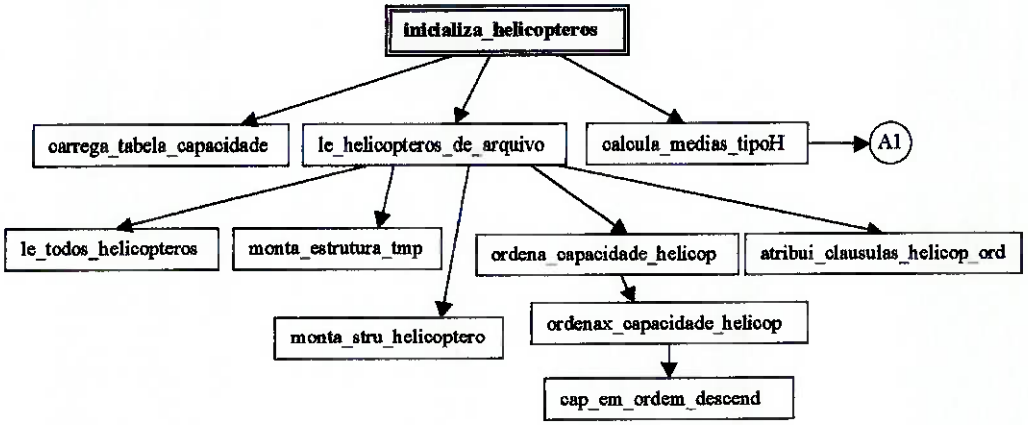


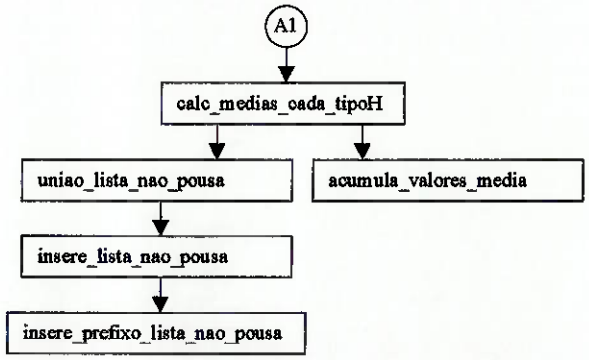
2) HELICOP (Obs.: ainda não implementada nesta versão)

3) HELICOP



4) HELICOP

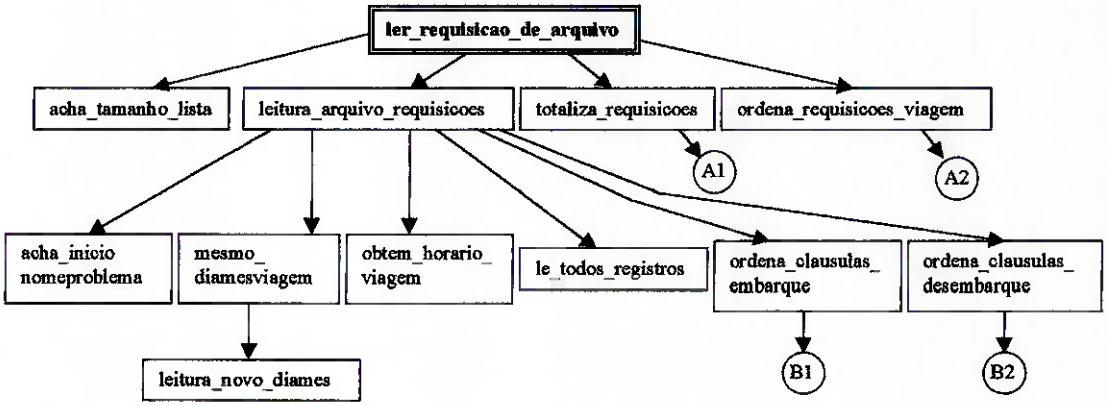


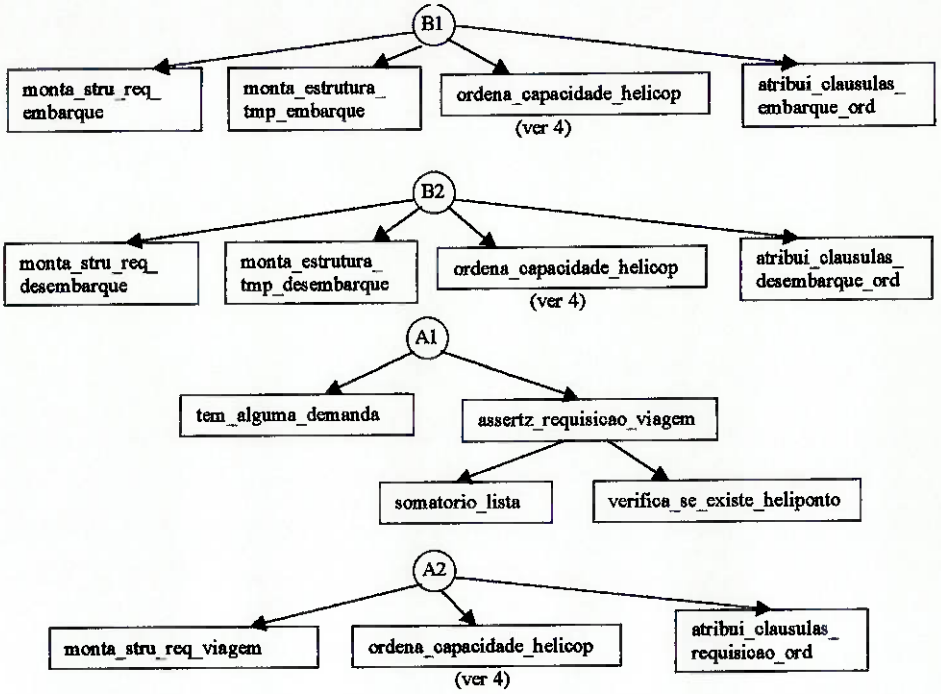


5) HELICOP

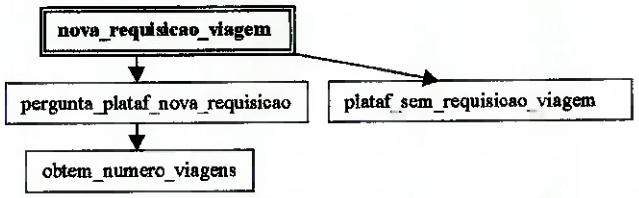
6) HELICOP

7) HELICOP

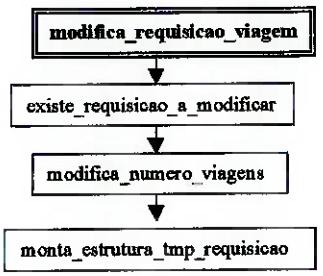




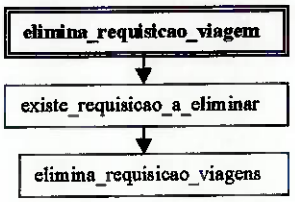
8) HELICOP



9) HELICOP



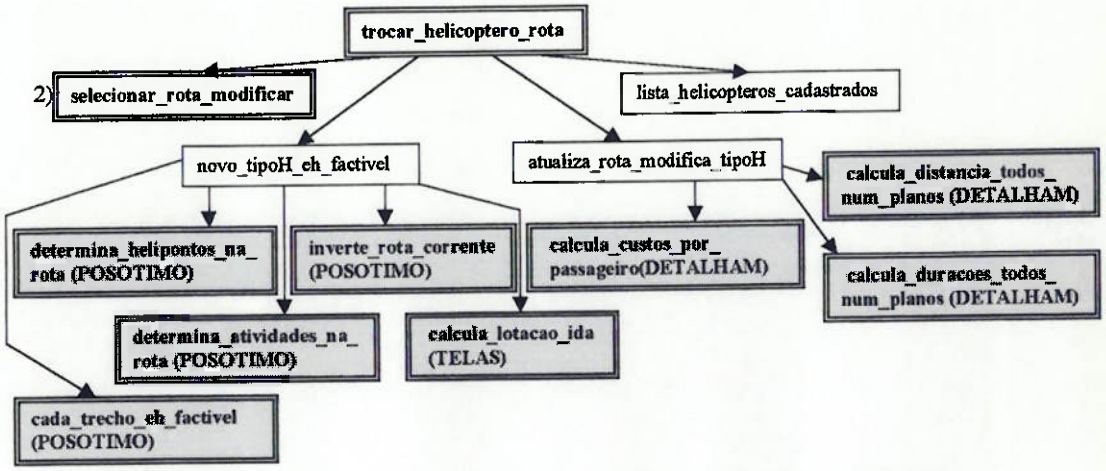
10) HELICOP



11) HELICOP

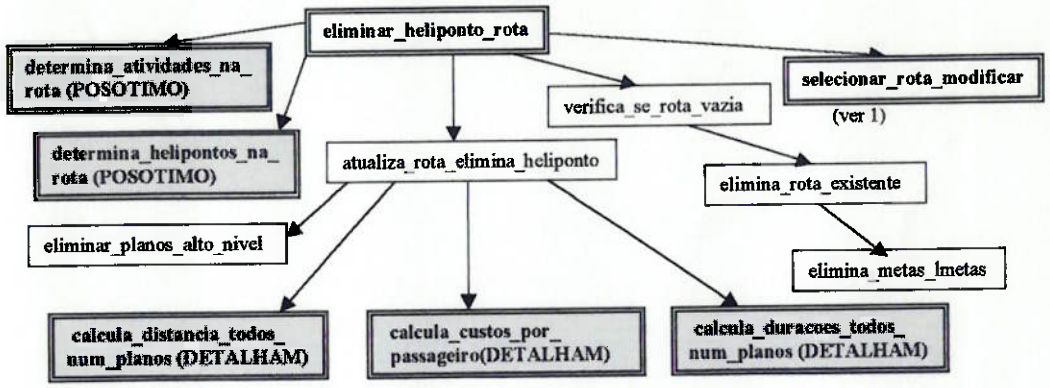
Arquivo: MODIFICA.PRO

1) HELICOP



2)

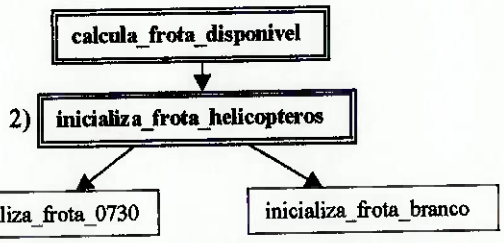
3) HELICOP



4) HELICOP (Obs.: ainda não implementada nessa versão)

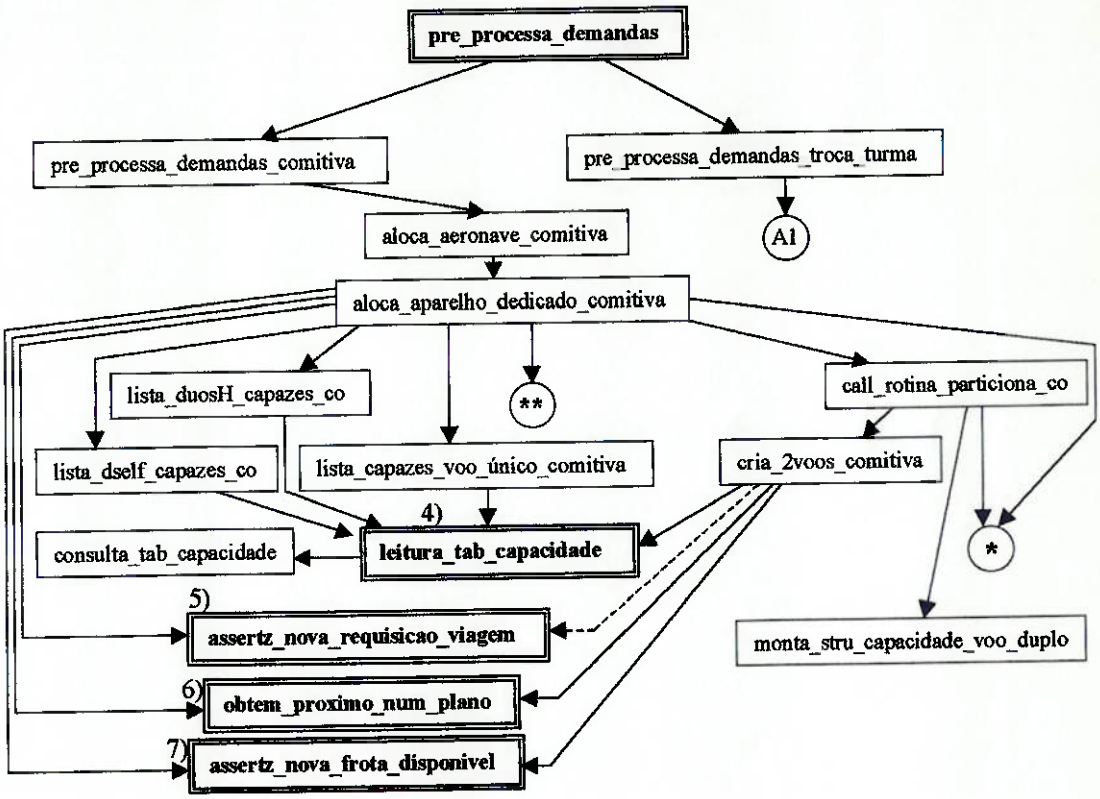
Arquivo: PREPRCSS.PRO

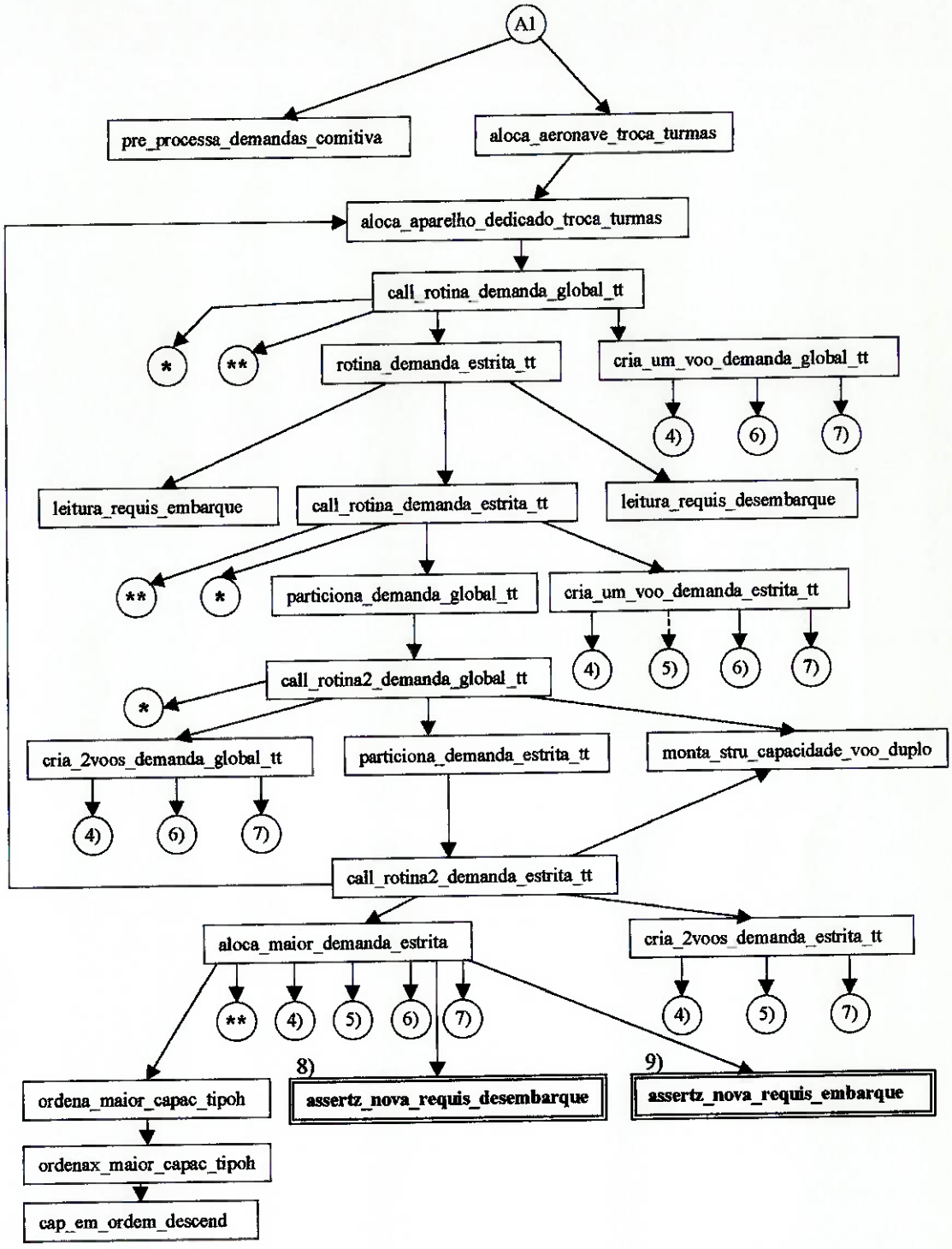
1) HELICOP / PLN_MNGR

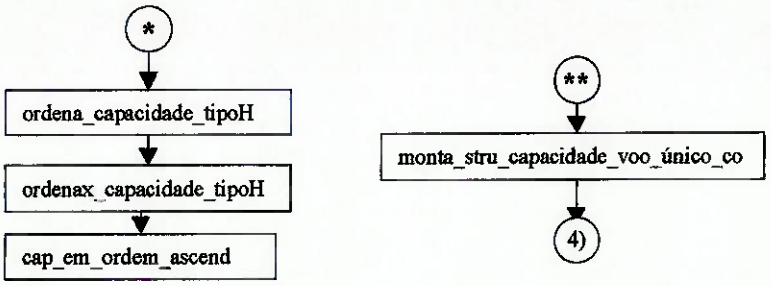


2)

3) PLN_MNGR




A1
pre_processa_demandas_comitiva
aloca_aeronave_troca_turmas
aloca_aparelho_dedicado_troca_turnas
call_rotina_demanda_global_tt
*
**
rotina_demanda_estrita_tt
cria_um_voo_demanda_global_tt
4)
6)
7)
leitura_requis_embarque
call_rotina_demanda_estrita_tt
leitura_requis_desembarque
**
*
particiona_demanda_global_tt
cria_um_voo_demanda_estrita_tt
4)
5)
6)
7)
call_rotina2_demanda_global_tt
*
cria_2voos_demanda_global_tt
particiona_demanda_estrita_tt
monta_stru_capacidade_voo_duplo
4)
6)
7)
call_rotina2_demanda_estrita_tt
aloca_maior_demanda_estrita
cria_2voos_demanda_estrita_tt
**
4)
5)
6)
7)
4)
5)
7)
8)
9)
ordena_maior_capac_tipoh
assertz_nova_requis_desembarque
assertz_nova_requis_embarque
ordenax_maior_capac_tipoh
cap_em_ordem_descend



---

4) PREGRUPO

5) PREGRUPO

6) PREGRUPO / PLN_GEN2

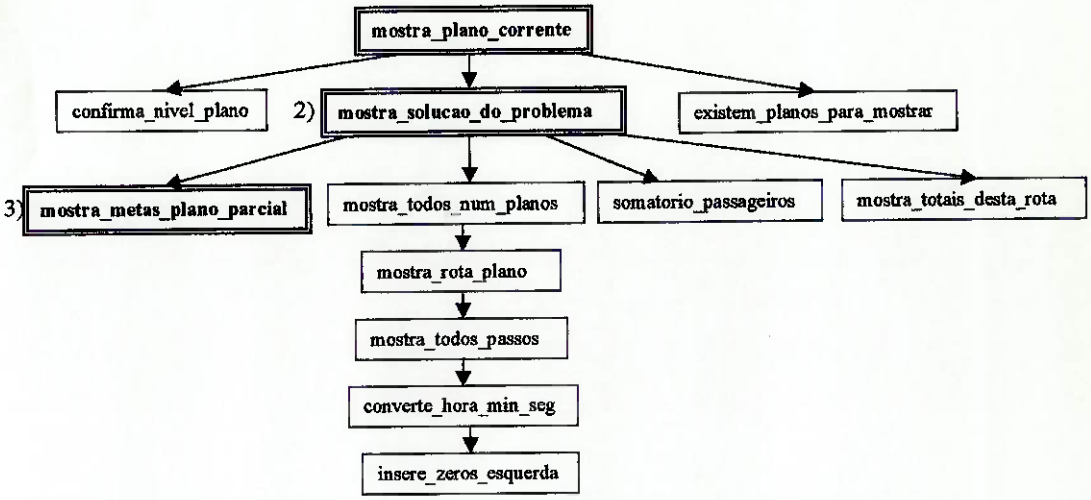
7) PREGRUPO

8) PREGRUPO

9) PREGRUPO

Arquivo: TELAS.PRO

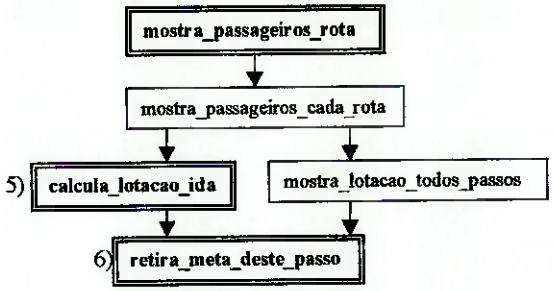
1) HELICOP



2) (PLN_GENE)

3)

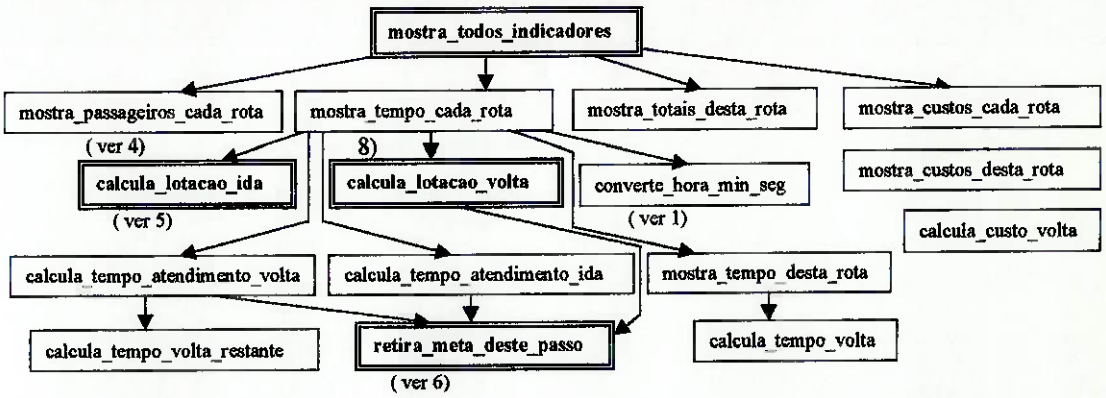
4) HELICOP



5) MODIFICA / POSOTIMO / PLN_GEN2 / DETALHAM

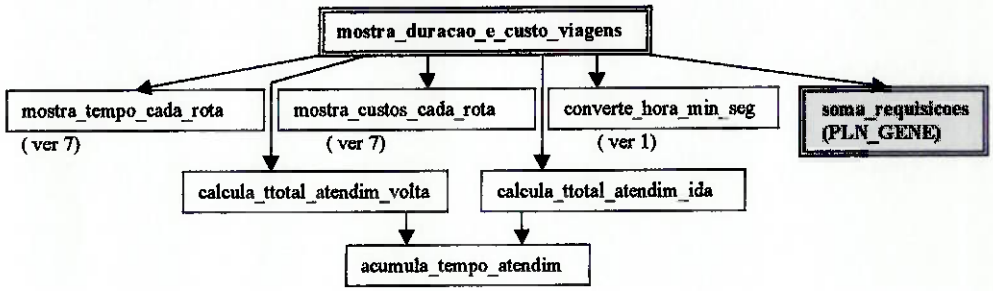
6) POSOTIMO / PLN_GEN2 / DETALHAM

7) HELICOP

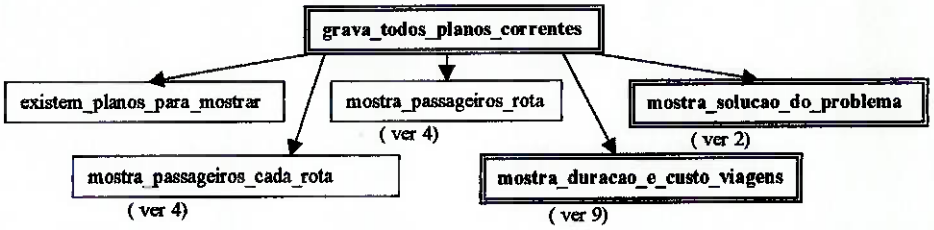


8)

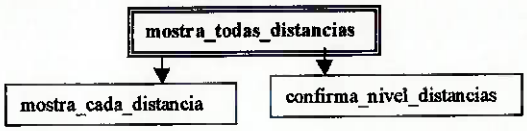
9) HELICOP



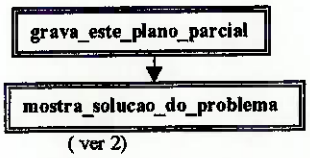
10) HELICOP



11) (HIERARQU)

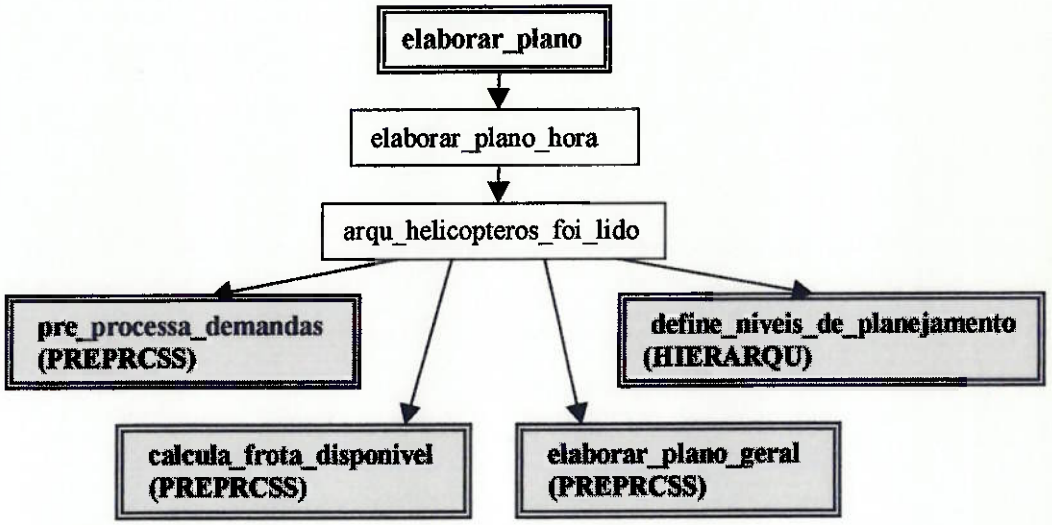


12) (PLN_GEN2)



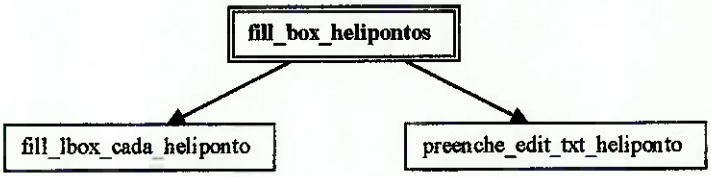
Arquivo: PLN_MNGR.PRO

1) HELICOP



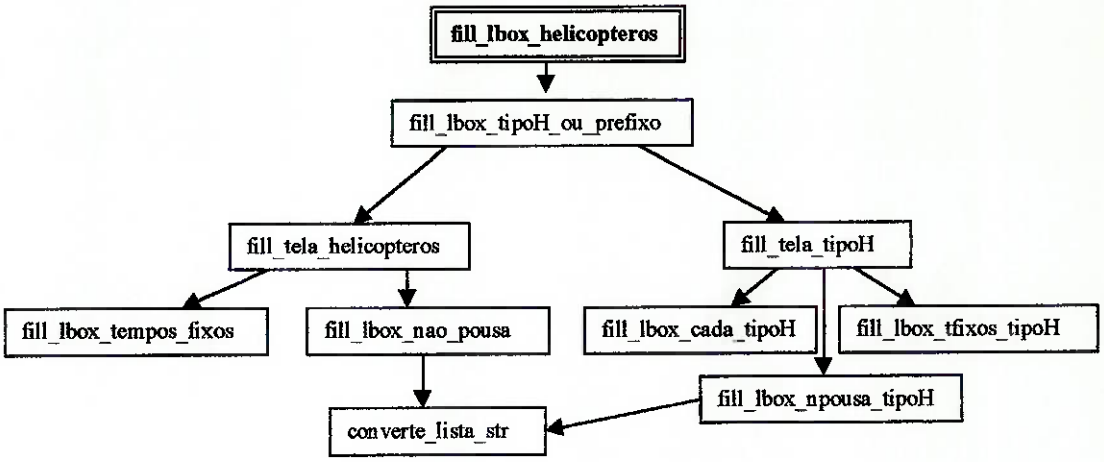
Arquivo: LBOX_STA.PRO

1) HELICOP

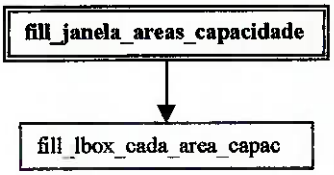


2) HELICOP

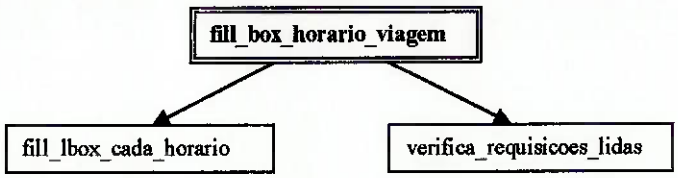
3) HELICOP



4) HELICOP

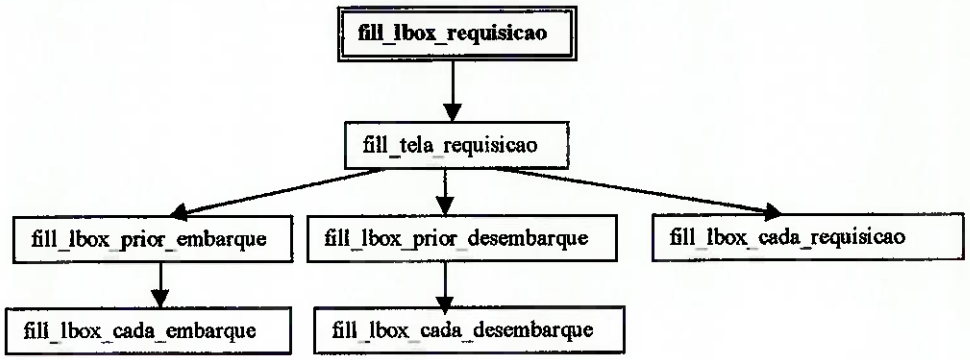


5) HELICOP



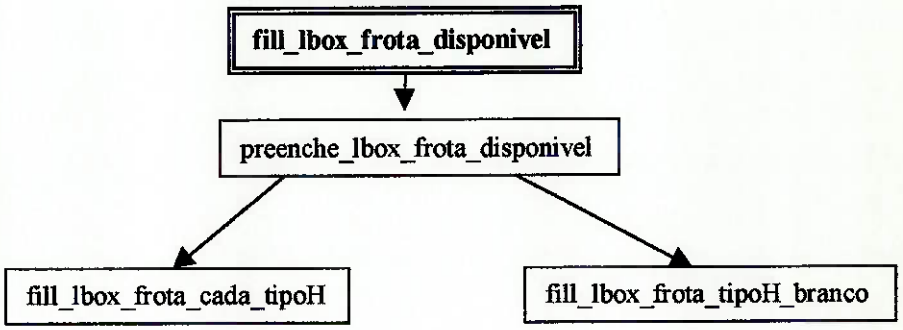
6) HELICOP

7) HELICOP



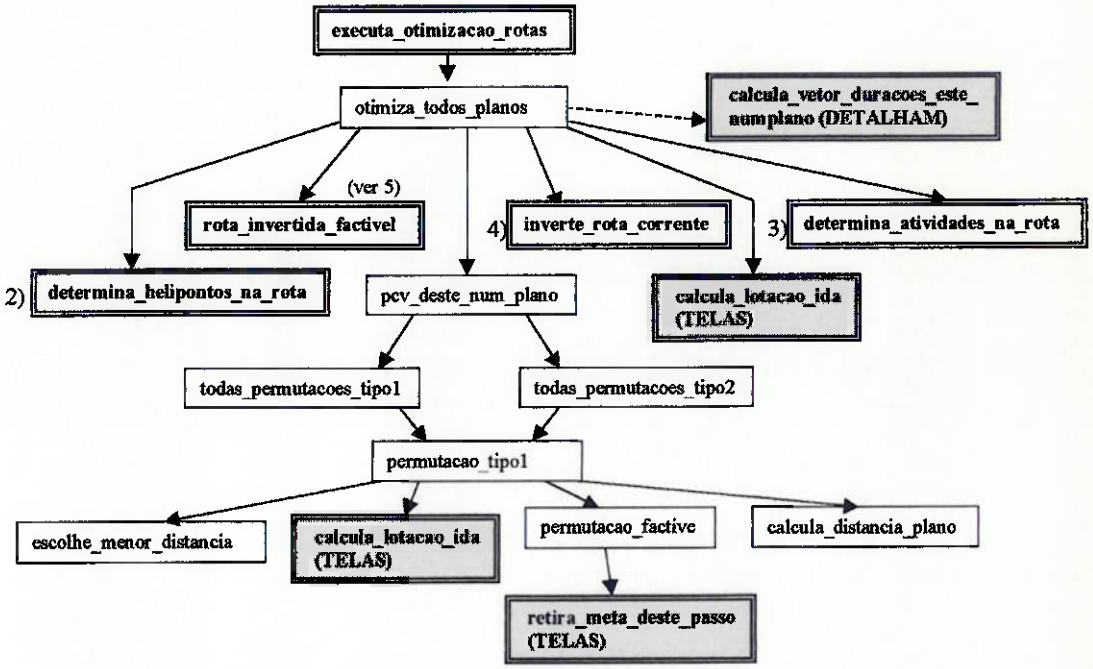
Arquivo: LBOX_PLN.PRO

1) HELICOP



Arquivo: POSOTIMO.PRO

1) (PLN_GEN2) / DETALHAM



---

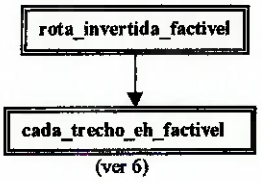
2) MODIFICA / DETALHAM

3) MODIFICA / DETALHAM

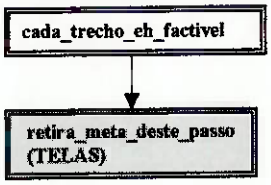
4) MODIFICA / DETALHAM

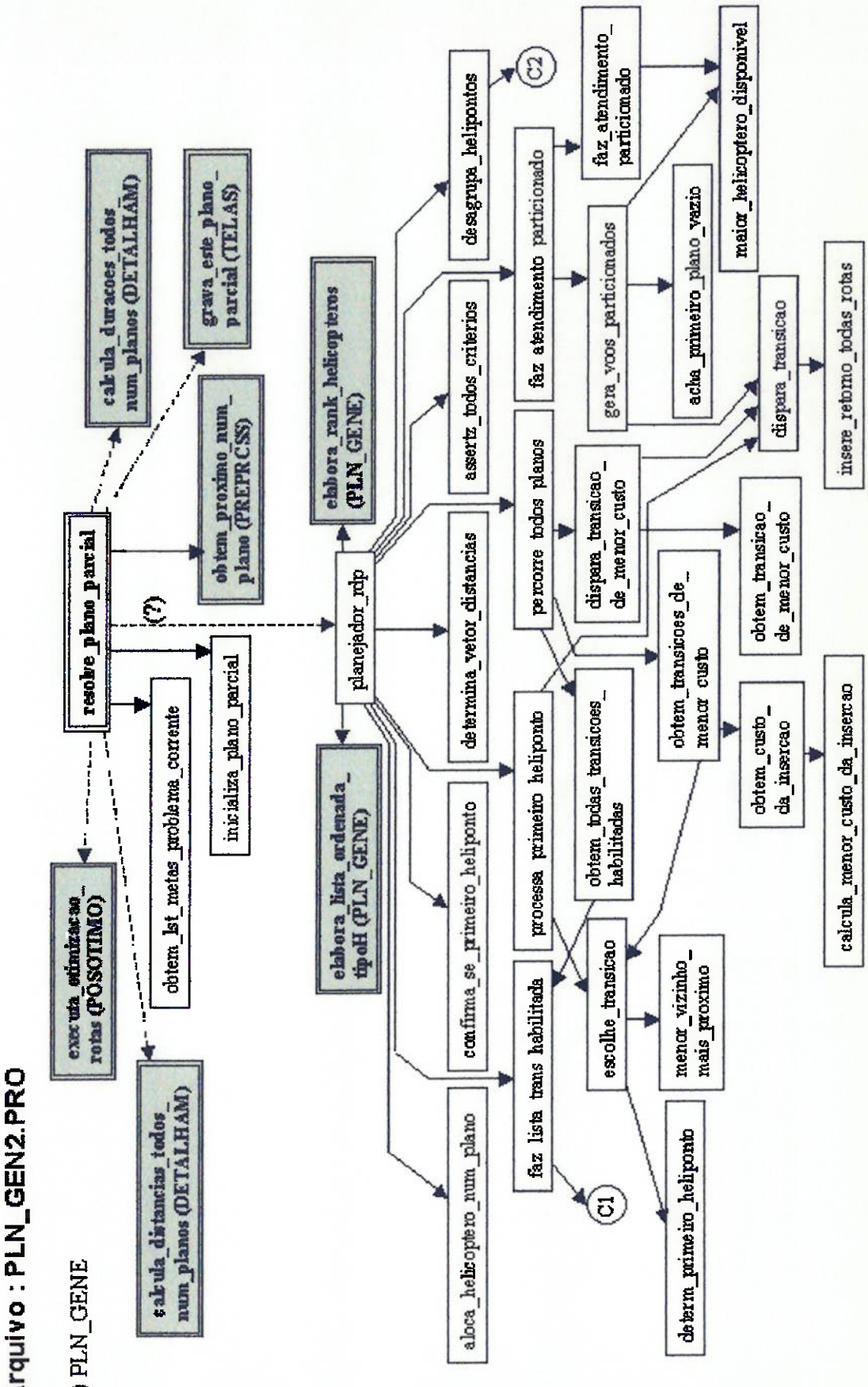
---

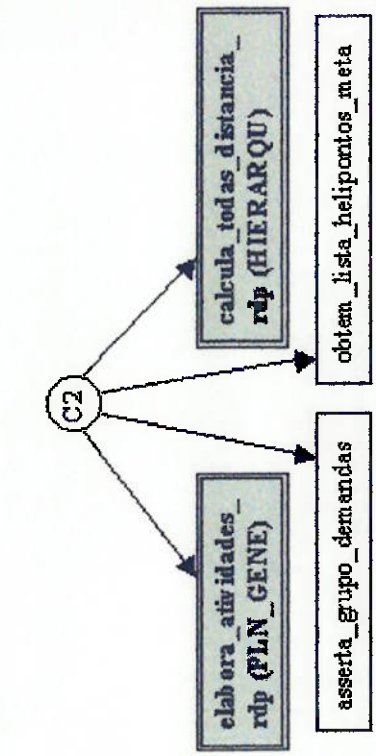
5) DETALHAM

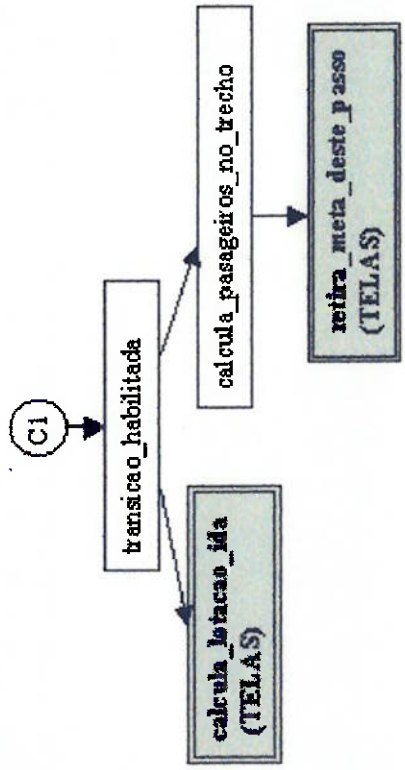


6) MODIFICA



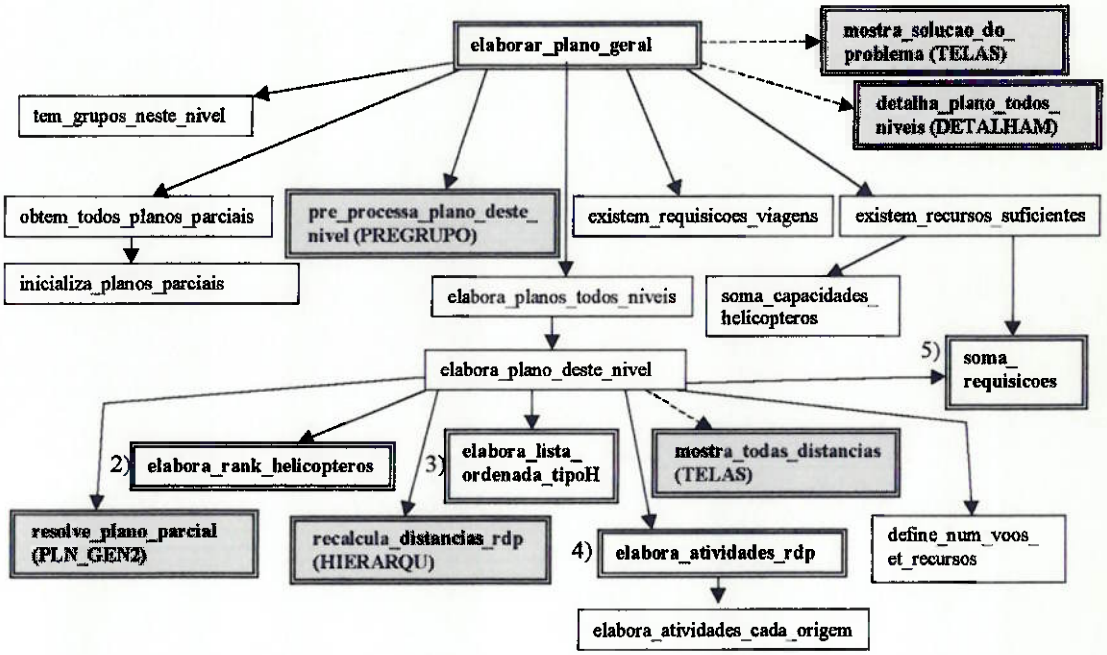

Arquivo : PLN_GEN2.PRO
1) PLN_GENE
executa_otimizacao_rotas (POSOTIMO)
resolve_plano_parcial
calcula_distancias_todos_num_planos (DETALHAM)
obtem_lst_metas_problema_corrente
inicializa_plano_parcial
(?)
calcula_duracoes_todos_num_planos (DETALHAM)
obtem_proximo_num_plano (PREPRCSS)
grava_este_plano_parcial (TELAS)
elabora_lista_ordenada_tipoH (PLN_GENE)
planejador_rdp
elabora_rank_helicopteros (PLN_GENE)
aloca_helicoptero_num_plano
confirma_se_primeiro_heliponto
determina_vetor_distancias
asserta_todos_criterios
desagrupa_helipontos
C2
faz_atendimento_particionado
faz_atendimento_particionado
percorre_todos_planos
processa_primeiro_heliponto
dispara_transicao_de_menor_custo
gera_voos_particionados
faz_atendimento_particionado
faz_lista_trans_habilitada
obtem_todas_transicoes_habilitadas
acha_primeiro_plano_vazio
maior_helicoptero_disponivel
C1
escolhe_transicao
obtem_transicoes_de_menor_custo
obtem_transicao_de_menor_custo
dispara_transicao
menor_vizinho_mais_proximo
obtem_custo_da_insercao
insere_retorno_todas_rotas
determ_primeiro_heliponto
calcula_menor_custo_da_insercao


C2
calcula_todas_distancia_
rdp (HIERARQU)
obtem_lista_helipontos_meta
elabora_atividades_
rdp (PLN_GENE)
asserta_grupo_demandas


C1
transicao_habilitada
calcula_passageiros_no_trecho
retira_meta_deste_passo
(TELAS)
calcula_latacao_ida
(TELAS)

Arquivo: PLN_GENE.PRO

1) PLN_MNGR



2) PLN_GEN2

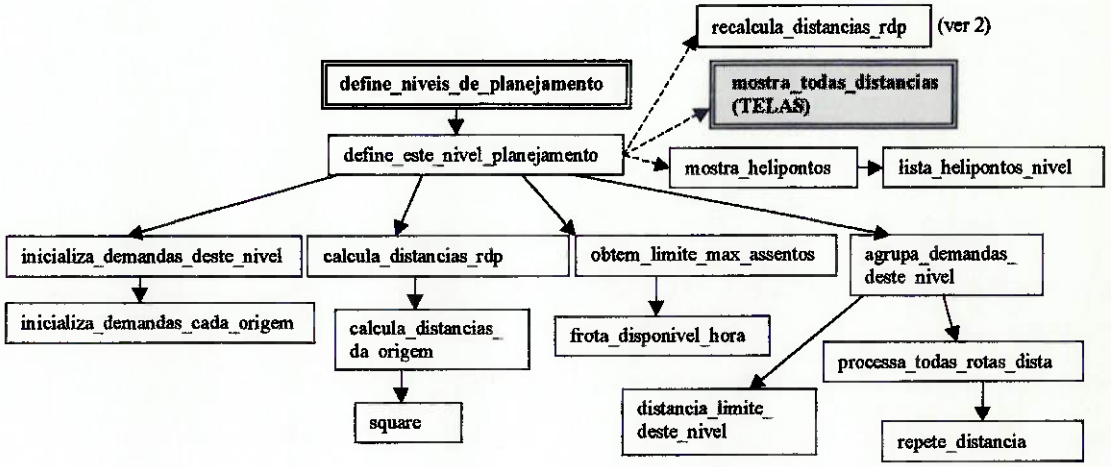
3) PLN_GEN2

4) PLN_GEN2 / DETALHAM

5) TELAS

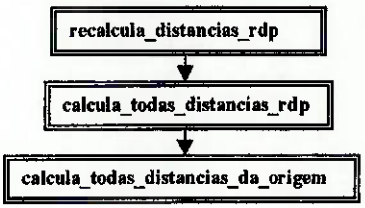
Arquivo: HIERARQU.PRO

1) PLN_MNGR



---

2) PLN_GENE / DETALHAM

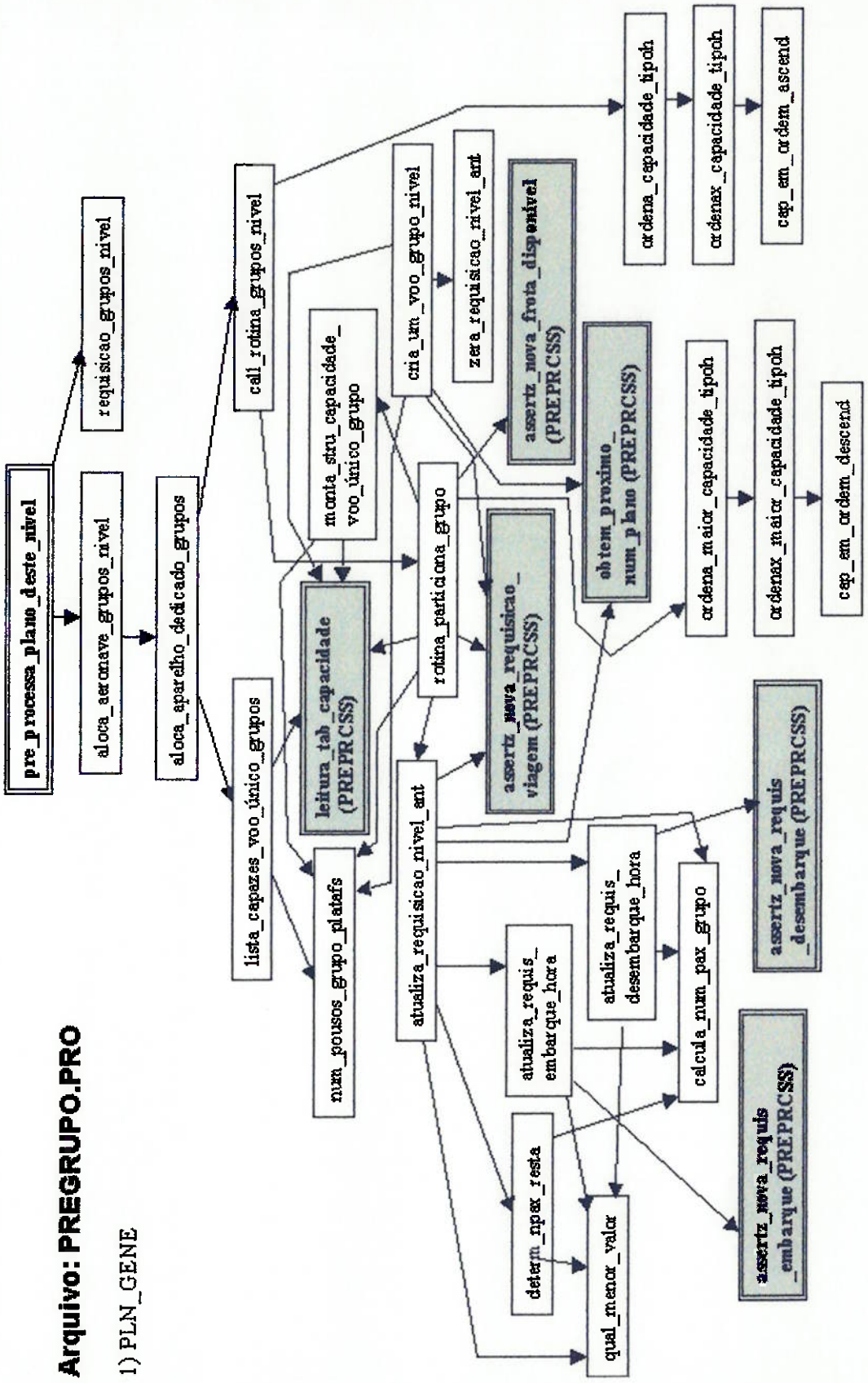


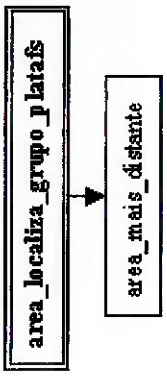
3) PLN_GEN2

4)

Arquivo: PREGRUPO.PRO

1) PLN_GENE

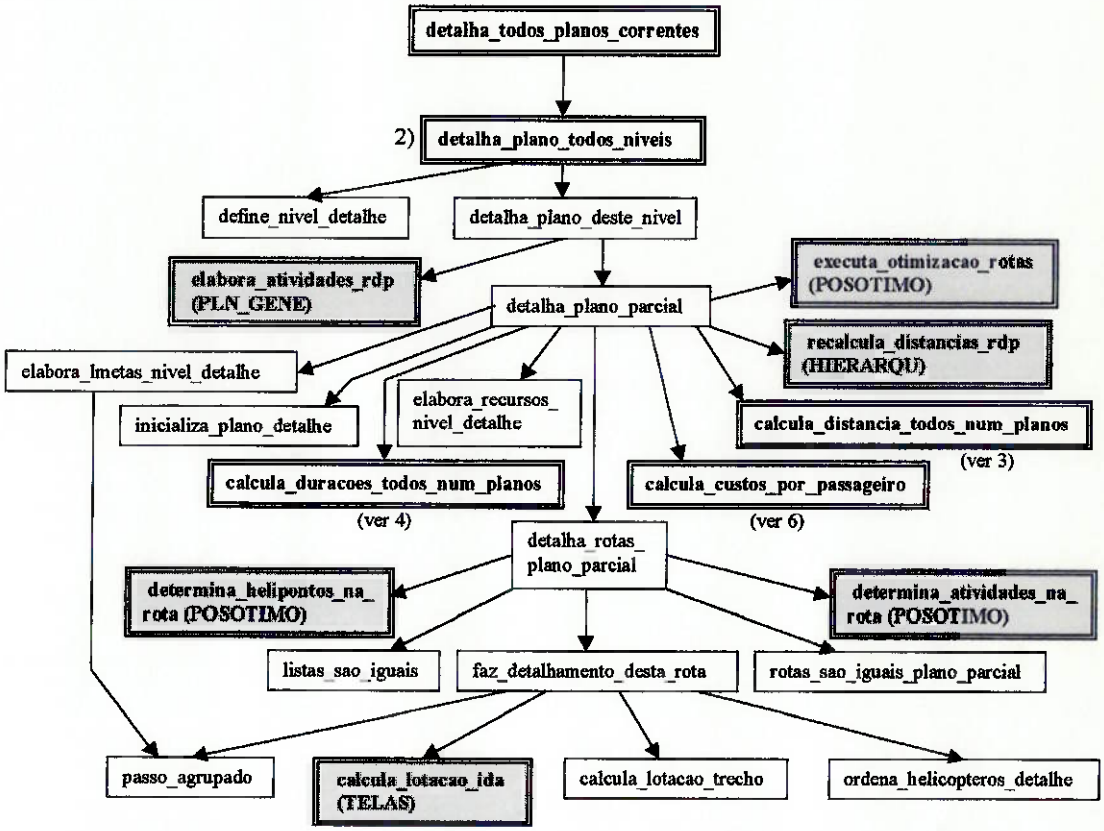




2)

Arquivo: DETALHAM.PRO

1)

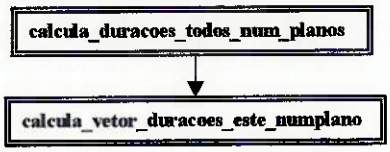


2) (PLN_GENE)

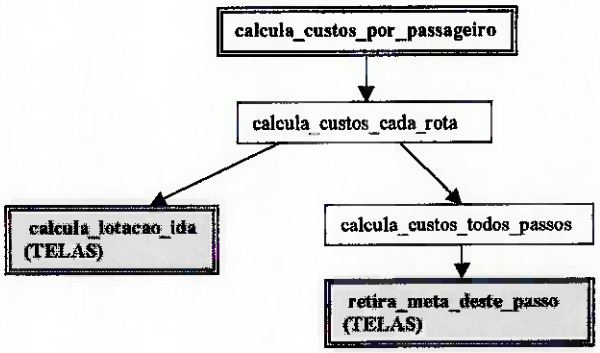
3) (PLN_GEN2)

4) (PLN_GEN2)

5) (POSOTIMO)



6)



11 Teste funcional de cada um dos ítens do menu

Ordem	Item	Resultado
1 e 6	*Arquivo* | | *Helipontos* | | *Ler_Arquivo_Helipontos*	Mostra janela p/ seleção de arquivo *.hpt. Selecionado: helipont.hpt. OK. Mensagem: "Helicopteros_Inicialização / Terminou leitura de helipontos". OK. Obs: apertando Cancel na seleção de arquivo, ocorre erro: Prolog ERROR 7002
4 e 8	*Arquivo* | | *Helipontos* | | *Incluir_Novo_Heliponto*	Mensagem: "Ainda não disponível". OK.
5	*Arquivo* | | *Helipontos* | | *Eliminar_Heliponto*	List_box c/ todos os helipontos. Selecionado: bgl1. Confirma: sim. Obs: verificou-se depois que realmente foi eliminado o heliponto da lista
2	*Arquivo* | | *Helicopteros* | | *Ler_Arquivo_Helicopteros*	Abre janela p/ seleção de arquivos *.hlp. Selecionado: helicop.hlp. Mensagem: "Terminou leitura de helicopteros"

Ordem	Item	Resultado
29	*Arquivo* | | *Helicopteros* | | *Eliminar_Helicoptero*	Pede o tipo de helicoptero a eliminar (não há list_box, é preciso saber o nome de cor). Oferecido b_XXX. Colocado b_212. OK. Obs: verificou-se depois que não se alterou a tela de helicopteros disponíveis.
3 e 7	*Arquivo* | | *Requisicoes_Viagens* | | *Ler_Arquivo_Requisicoes*	Abre janela p/ seleção de arquivos *.req. Selecionado: 2005730.req. Mensagem: "Terminou leitura do problema 2005730". Mesmo título.
10	*Arquivo* | | *Requisicoes_Viagens* | | *Nova_Requisicao* ***Botão Entra Nova Requisição***	List_box c/ helipontos p/ seleção de plataforma de origem / destino. Selecionada "ns16". Pede nº de passageiros ida: 2. Pede volta: 2. Mensagem: "Helicopteros_Inicialização / criada requisicao_viagem (ns16,2,2)"
11	*Arquivo* | | *Requisicoes_Viagens* | | *Modifica_Requisicao* ***Botaõ Edita Requisição***	Mostrada lista de helipontos c/ requisições (em ordem não alfabética). Selecionado "ns16". Pede nº passageiros ida: 1. Pede volta: 1. Pede confirmação. Mensagem: "modificada requisicao_viagem (ns16,1,1)"

Ordem	Item	Resultado
30	*Arquivo* | | *Requisicoes_Viagens* | | *Elimina_Requisicao*	list_box de requisicoes. Selecionada ns16. Confirma. Obs: foi verificado que a requisição realmente foi eliminada
34	*Sair*	Sair do programa
9	*Roteador* | | *Elaborar_Plano* ***Botão Elabora Plano***	Mensagens "Pre-processamento do plano / inicializa frota disponivel 7:30", "Elaborar plano / frota disponível for calculada", "Elaborar plano / pre-processamento foi calculado", "Elaborar plano / Hierarquia de Planos". Termina c/ PROLOG ERROR 1010
31	*Roteador* | | *Modificar_Plano_Existente* | | *Trocar_Helicopteros*	Pede nota a modificar: 1. Nada acontece.
32	*Roteador* | | *Modificar_Plano_Existente* | | *Eliminar_Heliponto_Rota*	Pede nota a modificar: 1. Nada acontece.
33	*Roteador* | | *Modificar_Plano_Existente* | | *Incluir_Heliponto_Rota*	Mensagem: "Helicopteros_Modifica Rota / incluir heliponto na rota". Nada acontece.

Ordem	Item	Resultado
12	*Telas* | | *Problema_Corrente* | | *Mostrar_ Helipontos*	Mostra list_box c/ todos os helipontos e suas coordenadas (no nível 0). No nível 1, em modo de coordenadas cartesianas, não mostra nada. C/ coordenadas UTM, mostra em ambos.
13	*Telas* | | *Problema_Corrente* | | *Mostrar_ Helicopteros*	Mostra todas as características dos helicópteros, tempos de chegada e saída da base e helipontos onde não pousa. É possível também mostrar tudo por prefixo.
14	*Telas* | | *Problema_Corrente* | | *Mostrar_ Requisicoes* ***Botaõ Ver Requisição***	No modo resumo, é possível ver todos os helipontos c/ suas req. de ida e volta. Nos modos embarque e desembarque, é possível ver tb a prioridade. É possível alterar dia, hora, e até os textos, mas isso não causa nenhum efeito.
15	*Telas* | | *Frota_Disponivel* | | *Calc_Frota_Disponivel*	Nesse exemplo estava cinza (indisponível)
16	*Telas* | | *Frota_Disponivel* | | *Mostra_Frota_Disponivel*	Mostra frota disponivel apenas p/ o horario das 7:30 (apesar de haver espaço p/ outros). Mostra apenas o tipo do helicóptero.

Ordem	Item	Resultado
17	*Telas* | | *Mostrar_Plano* | | *Mostrar_Plano_Corrente* ***Botão Mostra Plano***	Pede confirmação p/ nível do plano: 0. Nada acontece.
18	*Telas* | | *Mostrar_Plano* | | *Mostrar_Indicadores* | | *Num_Passageiros*	Mensagem: "Helicóptero -- Passageiros na rota / não existe nenhum plano elaborado"
19	*Telas* | | *Mostrar_Plano* | | *Mostrar_Indicadores* | | *Duracao_e_Custos*	Mensagem: "Helicóptero -- Duração e Custos / disponível apenas plano detalhado". Nada acontece.
20	*Telas* | | *Mostrar_Plano* | | *Mostrar_Indicadores* | | *Todos_Indicadores*	Mensagem: "Helicóptero -- Duração e Custos / disponível apenas plano detalhado". Nada acontece.
21	*Telas* | | *Mostrar_Plano* | | *Gravar_Plano_Corrente*	Abre janela de salvamento de arquivo, sugerindo 2005730.rot como nome. OK. Mensagens: "não existe plano corrente para nível 0", "não existe nenhum passageiro na rota", "não existe nenhum plano elaborado", "disponível apneas p/ plano detalhado". Terminou a gravação do arquivo.
22	*Parametros* | | *Horario_Viagem*	Mostra data e hora (no caso 20/05 7:30)

Ordem	Item	Resultado
23	*Parametros* | | *Tab Capacidade*	Mostra todos os tipos de helicópteros e as respectiva capacidades p/ cada uma das regiões.
24	*Parametros* | | *Areas_Capacidade*	Mostra os helipontos pertencentes a cada uma das áreas.
25	*Parametros* | | *Reserva_Capacidade*	Pede novo número p/ Reserva de Capacidade: 2. Mensagem: "foi atribuído 2 p/ reserva de capacidade".
26	*Ajuda* | | *Contents*	Mensagem: "não foi possível localizar o arquivo helicop.hlp. Você deseja localizar? Sim. Não.". Colocado Sim. Selecionado Helicop.hlp. Mensagem: "Arquivo não é ajuda ou está corrompido."
27	*Ajuda* | | *Local*	Nada acontece
28	*Ajuda* | | *About*	Mensagem: "No description. Version 1.0. Copyright (c) My Company".

Observações:

- Após eliminar um dos helipontos e cancelar uma tentativa de ler heliponto, o programa cancelou os anteriormente lidos. Foi necessário ler novamente os arquivos Helipontos, Helicopteros e de requisição

12 Upgrade e debugging

A versão preliminar do software, desenvolvida em Visual Prolog 4.0 foi atualizada para que as modificações pudessem ser implementadas no Visual Prolog 5.0. Para converter o projeto Helicop.VPR do Visual Prolog 4.0 para o 5.0 foi necessário:

- no prompt do DOS, rodar o programa de conversão UPGRADE.EXE, incluído na versão 5.0 do Visual Prolog:

  \VIP\UPGRADE\UPGRADE -i -d -pC:\HELICOP

  sendo C:\HELICOP o diretório que contém o projeto Helicop.VPR
- no ambiente do Visual Prolog 5.0, abrir o projeto Helicop.VPR (menu Project → Open Project...) e clicar no botão OK ao surgir a mensagem de alerta
- acionar o menu Options → Project (.VPR) → Application Expert CTRL+A, na tela Target selecionar a opção Platform em "Windows 32" e clicar no botão OK
- acionar novamente o menu Options → Project (.VPR) → Application Expert CTRL+A, na tela Target selecionar a opção Platform em "Windows 16" e clicar no botão OK, para gerar novamente os scripts de compilação do projeto
- acionar o menu Options → Project (.VPR) → Code Generator, na opção Options for Help Maker, desabilitar a caixa "Generate .HLP files and HLPTOPICS.CON files", para evitar a mensagem de erro "Invalid Command Call <HC31.EXE>" ao executar o projeto
- Fechar o projeto (menu Project → Close Project)
- Abrir novamente o projeto

- Abrir o arquivo Helicop.PRE (menu File → Open...) e modificá-lo deletando as linhas 18 a 20, para retirar as declarações duplicadas existentes nestas linhas
- Rodar o projeto (menu Project → Run)

Além disso, o programa original continha um "bug" que impedia a elaboração das rotas quando as plataformas a serem visitadas distavam mais de 12 km entre si, ou seja, quando não havia distinção entre os três níveis hierárquicos definidos (níveis 0, 1 e 2). Desse modo, o "bug" impedia a determinação das rotas quando somente o plano de nível 0 deveria ser elaborado. Para corrigir este problema, foi preciso:

- incluir o seguinte "catcher" para o predicado `define_este_nivel_planejamento` do módulo HIERARQU.PRO:

```
define_este_nivel_planejamento(0):-

        findall(Heliponto,todas_requisicoes_nivel(NivelAnt,Heliponto),LHelipontos
),
                eliminar_elementos_repetidos(LHelipontos,LHelipontosSR),
                retractall(distancia(0,_,_,_)),
                calcula_distancias_rdp(0,LHelipontosSR,LHelipontosSR),

                retractall(grupos_deste_nivel(0,_)),
                assertz(grupos_deste_nivel(0,[])),
                retractall(grupo_demandas(0,_,_,_)),

                heliponto(0,subrede_A,mcae,Abcissa,Ordenada),
                assertz(heliponto(0,subrede_AB,mcae,Abcissa,Ordenada)),
                recalcula_distancias_rdp(0),
                                !.
```

- modificar a chamada do predicado `define_niveis_de_planejamento` feita pelo predicado `elaborar_plano` no módulo PLANO.PRO:

```
define_niveis_de_planejamento(-1,NumNiveis),
```

- **incluir o seguinte "catcher" para o predicado** `elabora_planos_todos_niveis` **do módulo PLANO.PRO:**

```
elabora_planos_todos_niveis(0):-
            retractall(nivel_problema_corrente(_)),
            assertz(nivel_problema_corrente(0)),
            elabora_plano_subrede_a,
            elabora_plano_subrede_b,
            elabora_plano_subredes_ab,        !.
```

13 Modificações implementadas

As modificações do software original incluem a definição de pesos para as requisições de viagens, a redefinição do agrupamento das subredes de helipontos e o desenvolvimento da interface gráfica.

13.1 Requisições de viagem por peso transportado

O software foi modificado de forma que as requisições de viagem passaram a especificar o peso a ser transportado na ida e na volta para cada plataforma, ao invés do número de passageiros. Isso porque a limitação de transporte que deve ser considerada para cada helicóptero é, na verdade, a capacidade de carga do aparelho. Sendo assim, cada passageiro passou a ser considerado como uma carga (em kg) com o valor do peso da pessoa. Uma das vantagens da conversão é que, desta forma, o sistema Roteador pode tratar igualmente tanto as "perturbações" ao plano corrente com respeito a passageiros assim como com respeito às cargas.

Os arquivos *.RDP, que contêm as requisições de viagem tiveram seu formato modificado:

requisicao_viagem(Nivel,Heliponto,Ida,Volta)

Nivel = 0 (nível de abstração)

Heliponto = nome do heliponto

Ida = peso na viagem de ida (em kg)

Volta = peso na viagem de volta (em kg)

Segue abaixo a listagem de um exemplo de arquivo .RDP modificado (P0612AM.RDP):

```
requisicao_viagem(0,"pch1",200,100)
requisicao_viagem(0,"pna1",200,300)
requisicao_viagem(0,"ppg1",200,100)
requisicao_viagem(0,"pvm2",100,300)
requisicao_viagem(0,"ss20",400,300)
requisicao_viagem(0,"reel",300,200)
horario_viagem("08:00")
```

Para implementar essa modificação, todo o programa teve que ser revisto e diversas substituições foram feitas ao longo do código. Como um exemplo de predicado modificado, pode-se ver abaixo a nova listagem do predicado `elabora_rank_helicopteros` do módulo PLANO.PRO, em que passou a ser considerado o custo/(km.kg) de cada aparelho para elaborar o ranking por custo dos helicópteros.

```
%%=============================================================
%     ....   elabora rank de helicopteros disponiveis em termos de U$/km/kg
.......................
%%=============================================================
elabora_rank_helicopteros([],LCustos,LCustos):-  !.
elabora_rank_helicopteros([TipoH|LTipos],LParcial,LCustos):-
            helicoptero(TipoH,PUtil,Custo_HVoo,Veloc,_,_,_,_,_),
            CustoKmKg =   Custo_HVoo / (Veloc * PUtil),
            assertz(rank_helicopteros(TipoH,CustoKmKg)),
            concatena(LParcial,[CustoKmKg],LParcial1),
            elabora_rank_helicopteros(LTipos,LParcial1,LCustos).
```

13.2 Otimização de subredes

Conforme anteriormente citado, o modelo de domínio utilizado (e consequentemente o Application Model que dele se originou) contempla a divisão dos helipontos em sub-redes (vide Figura 6). A Figura 13 mostra a localização de todos os helipontos implementados no software.

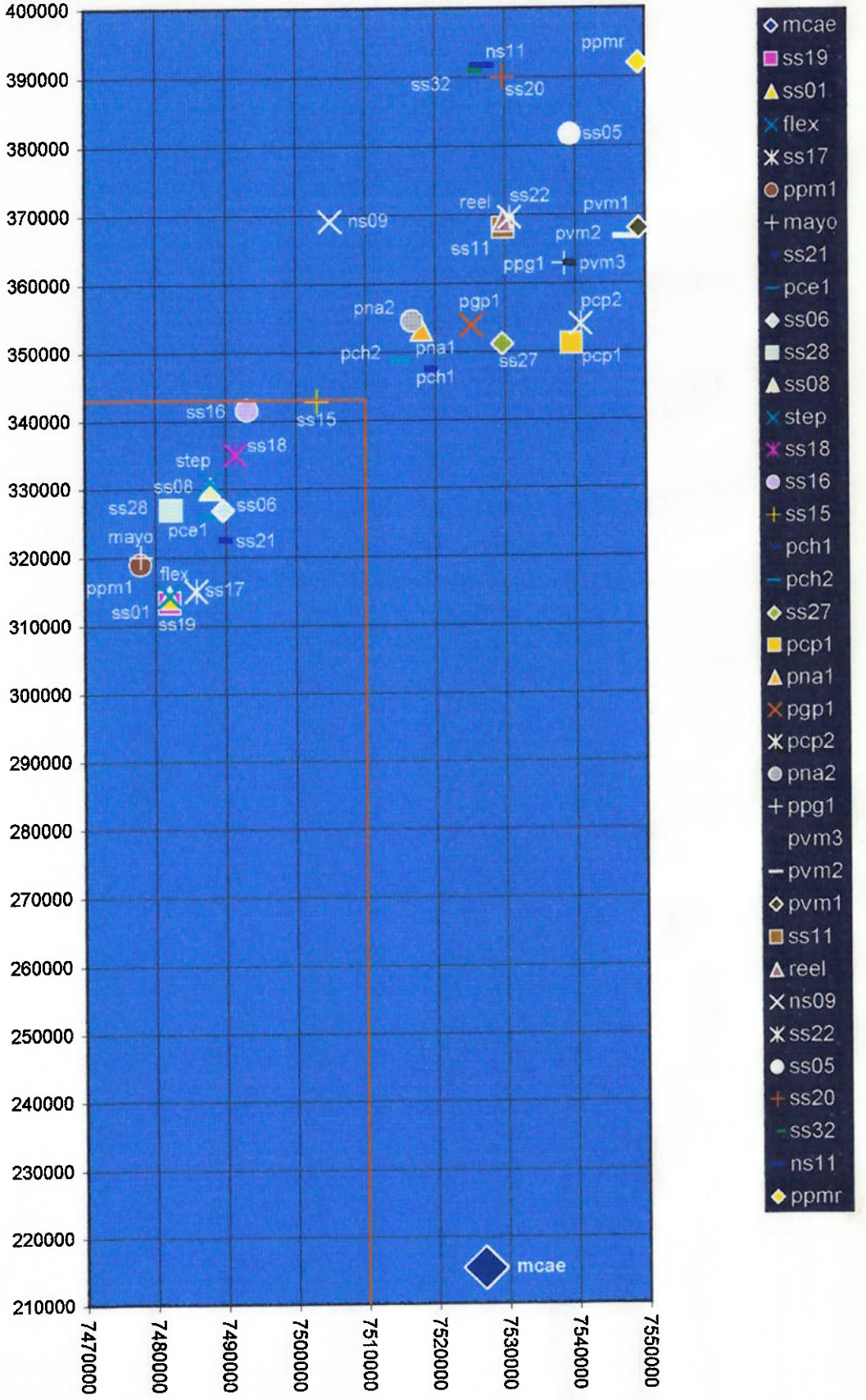


Figura 14 Localização geográfica dos helipontos implementados

A linha vermelha no gráfico da Figura 14 indica a divisão estática entre sub-redes A e B, de acordo com a definição implementada no programa original:

```
determina_subrede(StrOrdenada,StrAbcissa,subrede_A):-
        StrOrdenada <  343000,
        StrAbcissa <  7510000,
            !.
determina_subrede(_,_,subrede_B):- !.
```

A modificação implementada consiste na inclusão do predicado `otimiza_subredes` que verifica, dentre os helipontos a serem visitados, se existe algum cujo heliponto mais próximo esteja em outra sub-rede, mudando-o de sub-rede em caso afirmativo. Esta otimização dinamiza os limites das sub-redes, e evita que helipontos permaneçam "isolados" em suas sub-redes.

Para implementar essa modificação, o novo predicado `otimiza_subredes` foi incluído ao final do módulo MODIFICA.PRO:

```
%%=============================================================
%                        .....................          otimiza     subredes
.....................................
%%  Para cada heliponto que possui requisicao de viagem, verifica se heliponto
mais
%%  proximo pertence a mesma subrede e, se não pertencer, muda o heliponto de
subrede
%%=============================================================
otimiza_subredes(LHelipontos):-
          retractall(distancia(_,_,_,_)),
          calcula_todas_distancias_rdp(0,LHelipontos,LHelipontos),
          modifica_subredes(LHelipontos),
          retractall(distancia(_,_,_,_)).
%%%-------------------------------------------------------------
modifica_subredes([]):- !.
modifica_subredes([Heliponto|LHelipontos]):-
          heliponto(0,Subrede,Heliponto,_,_),
          findall(Dist,distancia(0,Heliponto,_,Dist),LDist),
          insertion_sort(LDist,LDist_Ord),
          LDist_Ord=[MenorDist|_],
          distancia(0,Heliponto,MaisProximo,MenorDist),
```

```
            heliponto(0,SubredeMP,MaisProximo,_,_),
            not(SubredeMP = Subrede),
            retract(heliponto(0,Subrede,Heliponto,Ordenada,Abcissa)),
            assertz(heliponto(0,SubredeMP,Heliponto,Ordenada,Abcissa)),
            modifica_subredes(LHelipontos).
modifica_subredes([_|LHelipontos]) :-      %% SubredeMP = Subrede
            modifica_subredes(LHelipontos).
```

Além disso, foi incluída a chamada do `otimiza_subredes` **no predicado** `elabora_plano`**:**

```
%%=============================================================
%%%%.........................                          PLANO        GLOBAL
.............................
%       Elabora o plano de alto nível, para o problema definido em INICIALI.PRO.
%       O nivel abordado é o de mais alto nivel, criado em HIERARQU.PRO.
%%=============================================================
elaborar_plano:-
         existem_requisicoes_viagens(LHelipontos),
         existem_recursos_suficientes(LHelipontos),
         nome_problema(NomeProblema),
         Msg="Defina o nome para o problema corrente. ",
         Title="Helicópteros  --  Elabora Plano",
         NovoNomeProbl=dlg_GetStr(Title,Msg,NomeProblema),
         retractall(nome_problema(_)),
         assertz(nome_problema(NovoNomeProbl)),
         otimiza_subredes(LHelipontos),
         define_niveis_de_planejamento(-1,NumNiveis),
         elabora_planos_todos_niveis(NumNiveis),          !,
         detalha_plano_todos_niveis,          !,
         subrede_corrente(0,Subrede),
         mostra_subredes_do_problema(0,NomeProblema,[Subrede]),          !.
```

São mostradas a seguir as saídas produzidas pelo software para um caso exemplo em que as sub-redes são modificadas:

```
subrede_a:
HELIPONTO ORDENADA  ABCISSA

     mcae  215308.92  7526587.82
     ss19  313432  7482033
     ss01  314151  7481909
     flex  314432  7482033
     ss17  315104  7485798
```

```
      ppm1   318940   7477927
      mayo   320000   7478000
      ss21   322501   7489220
      pce1   325941   7487934
      ss06   326876   7489597
      ss26   326965   7482255
      ss08   330052   7487815
      step   331052   7487815
      ss18   335055   7491324
      ss16   341535   7493073
      ss15   342811   7503009



subrede_b:
HELIPONTO ORDENADA  ABCISSA

      mcae   215308.92   7526587.82
      pch1   347501   7518651
      pch2   348744   7515020
      ss27   351147   7529606
      pcp1   351155   7539501
      pna1   353254   7518078
      pgp1   353939   7525159
      pcp2   354003   7540795
      pna2   354565   7516750
      ppg1   362828   7538445
      pvm3   362828   7538445
      pvm2   366700   7547000
      pvm1   368000   7549000
      ss11   368094   7529746
      ree1   369094   7529746
      ns09   369128   7504996
      ss22   369650   7530700
      ss05   381668   7539293
      ss20   390033   7529733
      ss32   391029   7524978
      ns11   391710   7526863
      ppmr   392002   7549012



Data: 9/12/1999      Hora: 16:28:29:70

 nivel: 0 problema: P0812AM horario: 08:00 subrede: subrede_ab

 metas:
```

```
heliponto:  ss28      ida: 200          volta:100
heliponto:  ss15      ida: 100          volta:300
heliponto:  pch2      ida: 400          volta:300
heliponto:  ss27      ida: 300          volta:200
heliponto:  ppm1      ida: 200          volta:100
heliponto:  flex      ida: 200          volta:300


 helicoptero: b-212 capac.:1683 kg
                     custo (U$/hr):   374.00  veloc.(km/hr):     185.2
nivel: 0  rota: 1            tempo de voo:    dista(km):

08:00:00     mcae-flex       00:35:12       108.68
08:35:12     flex-ppm1       00:01:58         6.10
08:37:11     ppm1-ss28       00:02:57         9.12
08:40:08     ss28-ss15       00:08:27        26.11
08:48:35     ss15-pch2       00:04:20        13.40
08:52:56     pch2-ss27       00:04:47        14.78
08:57:43     ss27-mcae       00:44:01       135.87
                                 Tempo total de Voo:01:41:44
                                           custo (U$):     634.21
                                           distancia:      314.054  km


                          custo total (U$):    634.21
                          distancia total:   314.054   km
                              peso total:  2700 kg
               custo unitario (U$/kg):      0.23


subrede_a:
HELIPONTO ORDENADA  ABCISSA

      mcae  215308.92  7526587.82
      ss19  313432  7482033
      ss01  314151  7481909
      flex  314432  7482033
      ss17  315104  7485798
      ppm1  318940  7477927
      mayo  320000  7478000
      ss21  322501  7489220
      pce1  325941  7487934
      ss06  326876  7489597
      ss28  326965  7482255
      ss08  330052  7487815
      step  331052  7487815
```

```
        ss18   335055   7491324
        ss16   341535   7493073



subrede_b:
HELIPONTO ORDENADA  ABCISSA

        mcae   215308.92   7526587.82
        pch1   347501   7518651
        pch2   348744   7515020
        ss27   351147   7529606
        pcp1   351155   7539501
        pna1   353254   7518078
        pgp1   353939   7525159
        pcp2   354003   7540795
        pna2   354565   7516750
        ppg1   362828   7538445
        pvm3   362828   7538445
        pvm2   366700   7547000
        pvm1   368000   7549000
        ss11   368094   7529746
        reel   369094   7529746
        ns09   369128   7504996
        ss22   369650   7530700
        ss05   381668   7539293
        ss20   390033   7529733
        ss32   391029   7524978
        ns11   391710   7526863
        ppmr   392002   7549012
        ss15   342811   7503009
```

13.3 Interface gráfica

Uma amostra da interface gráfica implementada pode ser vista na Figura 15.

Figura 15 O software Roteador de Helicópteros

14 Apêndice I - Trabalho apresentado no SICUSP 97

Aplicação de Redes de Petri a um Problema de Planejamento

Adriana Jacoto
Programa Especial de Treinamento - Eng. Mecatrônica
Orientador: Prof. Dr. José Reinaldo Silva
EPUSP

Abstract

This paper shows, using the Blocks World model-problem, how a Petri Net structure analysis provides the knowledge necessary to guide the planning process. Through the results of the Petri Net structure analysis, we may establish a strategy for the problem solution process. The Petri Net makes the problem modeling as a Discrete Event System and is able to capture the productive system's main features and its dynamic behavior.

1. Introdução

A Rede de Petri promove a modelagem do problema de Planejamento como um Sistema de Eventos Discretos (SED) e pode capturar a característica principal de um sistema produtivo que é o seu comportamento dinâmico. Como exemplo de aplicação, foi desenvolvida a estrutura da rede de Petri que representa o domínio do problema do Mundo de Blocos. Este problema consiste em se elaborar um plano para que um robô empilhe blocos, visando, a partir de uma dada configuração inicial, atingir um estado final previamente determinado. A partir deste desenvolvimento, pode-se realizar a simulação do sistema, utilizando-se programação em PROLOG aliada ao controle de um robô.

O estudo de ações e de planos é fundamental no desenvolvimento de sistemas inteligentes que sejam capazes de lidar eficazmente com os problemas do mundo real. No processo de planejamento, a decisão do planejador acerca de qual curso de ação tomar é escolhida a partir de um vasto repertório de possibilidades. Esta decisão, por sua vez pode influenciar os estados do mundo de maneiras diversas e complicadas. Tudo isto é realizado num mundo complexo e dinâmico em que o meio-ambiente está em mudança contínua.

Para se elaborar um sistema de "planning", é necessário um modelo de mundo e um modelo de ação para atuar neste modelo de mundo. O modelo de mundo pode ser aberto ou fechado. O modelo aberto trata do ambiente desestrururado onde o planejador não tem controle sobre todos os eventos que ocorrem no mundo. Por este motivo, este modelo é computacionalemente complexo. Para se reduzir o problema computacional associado ao modelo aberto, muitas vezes estabelece-se a hipótese do mundo fechado. Neste modelo, apenas as cláusulas cuja interpretação é conhecida podem ser verdadeiras. Quando é referenciada uma cláusula cuja interpretação não é conhecida, então, esta é imediatamente assumida como sendo falsa. Isto reduz muito o problema computacional, mas, em contrapartida, limita a aplicabilidade dos sistemas assim desenvolvidos.

A forma de representação dos eventos e ações é determinada pelo modelo de ação. Este define as formas possíveis para se atuar sobre o modelo de mundo. O modelo de mundo pode estar num dos seus infinitos estados. Um estado é uma foto instantânea do mundo em determinado instante.

Um plano pode ser visto como sendo a especificação de uma sequência de estados desejada sobre o modelo de mundo para se atingir uma determinada meta, a partir de um estado inicial conhecido. Através da análise das propriedades estruturais da Redes de Petri que representa o domínio da aplicação, podemos aprender estratégias de busca e dirigir o processo de resolução do plano.

2. O Mundo de Blocos

Inicialmente foi proposto o problema de se elaborar um plano para que um robô empurrasse três blocos de modo a juntá-los. Em 1974, Gerald Jay Sussman propôs uma variante deste problema que consiste em se empilhar blocos, que ficou conhecido como o problema do Mundo de Blocos. Este problema é importante porque, apesar da sua aparente simplicidade, ele

apresenta todas as características que dificultam a automação dos problemas de planejamento.

Os sistemas clássicos de planejamento definem o plano como uma sequência de operadores que representa ações que devem ser efetuadas para se transformar um determinado estado inicial no estado meta desejado. Em tais sistemas a elaboração do plano não leva em conta o sucesso ou falha na execução do plano. Os sistemas mais significativos dentre os sistemas clássicos de planejamento são o General Problem Solver (GPS) e o Stanford Research Institute Problem Solver (STRIPS).

A principal contribuição do GPS está na abordagem meios-fins, que constitui uma técnica para dirigir o processo de busca de soluções. Basicamente, a abordagem meios-fins é uma forma de atingir a meta estabelecendo submetas, cujo atingimento leva ao atingimento da meta inicial. No GPS, as sentenças lógicas são chamadas de objetos, que podem ser comparados com os estados do mundo.

O STRIPS foi o primeiro sistema de planejamento bem sucedido. Este sistema tenta encontrar uma sequência de operadores no espaço de modelos do mundo, para transformar um dado modelo do mundo inicial num modelo em que uma determinada meta possa ser provada como sendo verdadeira. O STRIPS representa um modelo do mundo com uma coleção arbitrária de fórmulas do cálculo de predicados de primeira ordem, empregando um provador de teoremas, que emprega o método de resolução, para responder a questões de determinados modelos e usa a análise meios-fins empregada no GPS para guiá-lo até o modelo do mundo desejado que satisfaça à meta.

A maioria dos sistemas de planejamento utiliza uma representação do tipo STRIPS. Nesta representação, a descrição de uma ação apresenta: uma pré-condição, uma lista de adição e uma lista de eliminação. A lista de adição define cláusulas que poderiam não ser verdadeiras no modelo original mas que são verdadeiras no novo modelo que resulta após a aplicação do operador. A lista de eliminação especifica cláusulas do modelo original que não são mais verdadeiras no novo modelo.

Busca-se atingir o modelo do mundo desejado aplicando-se sucessivamente um conjunto de operadores STRIPS a partir do modelo do mundo inicial. Os sistemas aplicam ações para modificar o estado do mundo se suas pré-condições forem verdadeiras. O efeito da aplicação de uma ação está expresso nas listas de adição e de eliminação.

Entretanto, esta forma de representação apresenta problemas, sendo a sua principal deficiência o fato de não considerar a dependência entre metas concorrentes. Isto porque adotou-se a teoria linear que assume que quaisquer duas submetas de um problema podem sempre ser obtidas de uma forma independente. Desta forma, uma conjunção de metas era dividida em submetas para as quais se tentava obter uma solução isoladamente. É claro que esta hipótese não é verdadeira para a maioria dos problemas de "planning". Por este motivo, um sistema que emprega este tipo de representação não é capaz de contextualizar corretamente o efeito da aplicação de uma ação.

Uma consequência importante desta deficiência é a Anomalia de Sussman. Na Anomalia de Sussman, o atingimento de algumas metas pode requerer o estabelecimento de submetas para satisfazer pré-condições que irão desfazer cláusulas do estado meta anteriormente já verdadeiras. Deste modo, o programa entra num processo de "looping" sem progresso em direção ao estado meta desejado.

Considerando-se o problema do Mundo de Blocos, suponha-se que existam três blocos na mesa, A, B e C, e que se deseja construir uma pilha onde A esteja sobre B, B sobre C, e este último sobre a mesa. Nenhuma conjunção é encontrada para resolver o problema. Então, é estabelecida uma estratégia de resolução. A teoria linear que determina as conjunções sugere que, primeiramente, mova-se A sobre B, e depois, B sobre C. Se as submetas são independentes, sua ordem não importa, então a ordem arbitrária é adotada. Desta forma, o robô primeiro coloca A sobre B. Em seguida, tenta colocar B sobre C, mas isto significa que ele necessita mover B. Mas o robô não pode mover B com A sobre ele (existe uma restrição física à mão do robô), e assim ele remove A de B e coloca A sobre a mesa. A seguir, coloca

B sobre C e considera que a meta foi atingida. Entretanto, A não se encontra sobre B.

Este fenômeno acontece porque no modelo de ação do STRIPS, o efeito da aplicação de cada ação não está bem contextualizado. Ações distintas no seu efeito são erroneamente avaliadas como equivalentes. Todos os operadores apresentam igual oportunidade de serem escolhidos na hora de se decidir qual operador será aplicado em seguida. A abordagem meios-fins, criada no GPS, é o único critério empregado para se discriminar os operadores. Se o operador for considerado relevante para o atingimento do estado meta e se todas as suas pré-condições forem verdadeiras, então, o operador é aplicado logo em seguida.

3. Redes de Petri

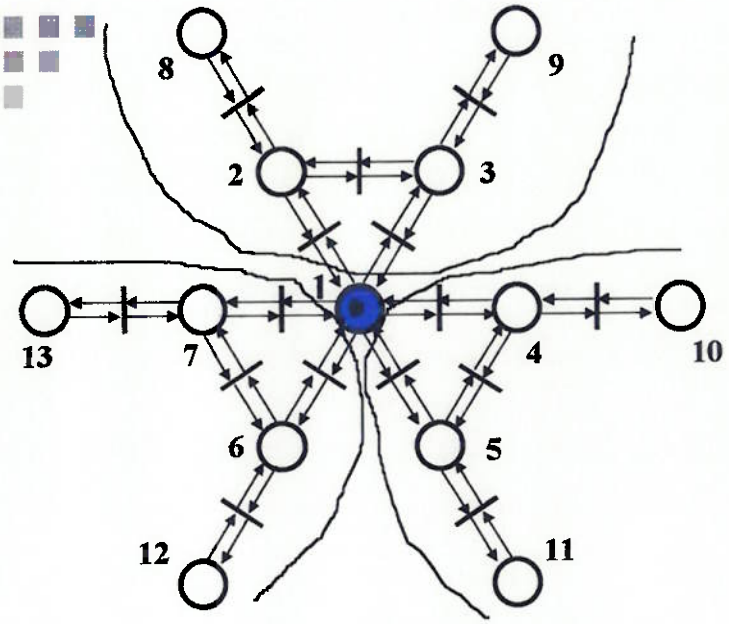
A análise da Rede de Petri que representa o domínio do problema pode sugerir estratégias de busca para dirigir o processo de solução do problema, por exemplo, para definir quais transições estão habilitadas em determinado instante.

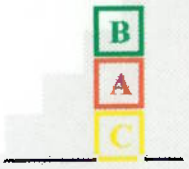
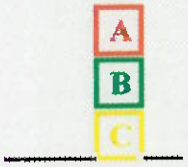
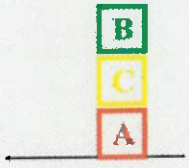
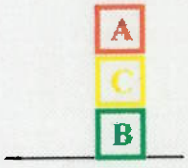
A teoria das Redes de Petri é aplicada tradicionalmente ao problema de controle de sistemas discretos. A Rede de Petri permite a representação formal dos estados e da evolução de um sistema de eventos discretos. Permite explicitar relações entre estados, por exemplo: dependência de ordem parcial entre estados, existência de eventos concorrentes ou excludentes, compartilhamento de recursos, etc.

A análise estrutural de uma Rede de Petri pode prover informação importante para uma abordagem não-procedimental em sistemas de eventos discretos. O conhecimento sobre a estrutura do problema pode subsidiar formas de busca mais eficazes, e a minimização do "backtracking".

A figura a seguir mostra a representação em Redes de Petri que constitui o modelo para o problema do Mundo de Blocos. A partir da figura, é possível observar-se que todos os estados podem ser atingidos a partir do estado em que todos os blocos estão sobre a mesa. Este estado é designado "nó essencial".

Além disso, pode-se identificar três componentes conectadas ou subredes. Estas subredes definem agrupamentos de ações inter-relacionadas pela causalidade. Estas subredes estão interligadas através do "nó essencial". Desta forma, analisando a estrutura da Rede de Petri, podemos efetuar uma decomposição do problema em subproblemas que são independentes, de uma



NÓ ESSENCIAL 1 			
2 	3 	8 	9 
4 	5 	10 	11 
6 	7 	12 	13 

maneira tal que:

- o estado onde todos os blocos estão sobre a mesa é um nó essencial do grafo
- o primeiro bloco movido a partir do nó essencial determina uma sucessão de estados dependentes desta ação que estão na mesma componente conectada
- existe sempre um plano que corresponde à distância mínima entre dois estados que pertencem a uma mesma componente conectada
- o plano envolvendo dois estados em duas componentes conectadas diferentes deve passar necessariamente pelo nó essencial

O conhecimento obtido a partir de uma análise da Rede de Petri que representa o domínio para este problema é muito importante. Como resultado da análise da Rede de Petri, pode-se sugerir o seguinte algoritmo para resolver este problema de planejamento:

Sejam dados *EI* = estado inicial, *M* = conjunção de metas e *NE* = nó essencial:

1) Se *M* já é verdadeiro (*EI* = *M*) então fim
2) Se *EI* e *M* estão na mesma componente conectada então existe um caminho entre *EI* e *M* sem passar pelo nó essencial. Resolva *M* usando um mecanismo do tipo meios-fins, obtendo o plano final *P*
3) Caso contrário, gere uma submeta *NE* = nó essencial obtendo um sub-plano *Parcial-1*. A seguir, resolva *M* a partir do nó essencial (*NE*) usando um mecanismo do tipo meios-fins obtendo um sub-plano *Parcial-2*. O plano final resulta da concatenação dos planos parciais *P* = *Parcial-1* + *Parcial-2*

O algoritmo usa o conhecimento obtido a partir de uma análise da Rede de Petri que representa o conhecimento do domínio e é capaz de evitar o problema da Anomalia de Sussman. Este algoritmo faz a decomposição do problema baseado na análise da estrutura da rede. Este conhecimento está baseado na noção de transições independentes para assegurar a independência de cada uma das subredes e na existência do "nó essencial" que faz a conexão entre todas estas subredes. A interdependência pela causalidade ocorre apenas entre ações que pertencem à mesma componente conectada do grafo

4. Conclusão

Posteriormente, o mesmo método poderá ser aplicado à automação de

uma célula de manufatura. Baseado no fato de que o processo de automatização é crescente, conclui-se que, torna-se cada vez mais importante obter metodologias para desenvolvimento destes sistemas de planejamento que assegurem a elaboração de planos com garantia da qualidade.

A aplicação da programação em PROLOG a esta área da inteligência artificial tem se revelado prática e eficiente, e tem sido utilizada no processo de implementação prática dos conceitos de planning então descritos. Com esse propósito, têm sido utilizados também dois robôs MOVEMASTEREX Mitsubishi, pertencentes ao LSI (Laboratório de Sistemas Integráveis) na Escola Politécnica da USP.

5. Bibliografia

SHIMADA, L.M. **Estruturação do Problema de Planejamento em uma Abordagem Baseada em IA e no Formalismo de Redes de Petri.** São Paulo, 1997.

SUSSMAN, G.J. **The Virtuous Nature of Bugs.** MIT AI Lab, 1974.

15 Apêndice II - Trabalho apresentado no SICUSP 98

Implementação em Visual PROLOG de um Roteador de Helicópteros

André de Bessa Santos
Adriana Jacoto
Programa Especial de Treinamento - Eng. Mecatrônica
Orientador: Prof. Dr. José Reinaldo Silva
EPUSP

1. Introdução

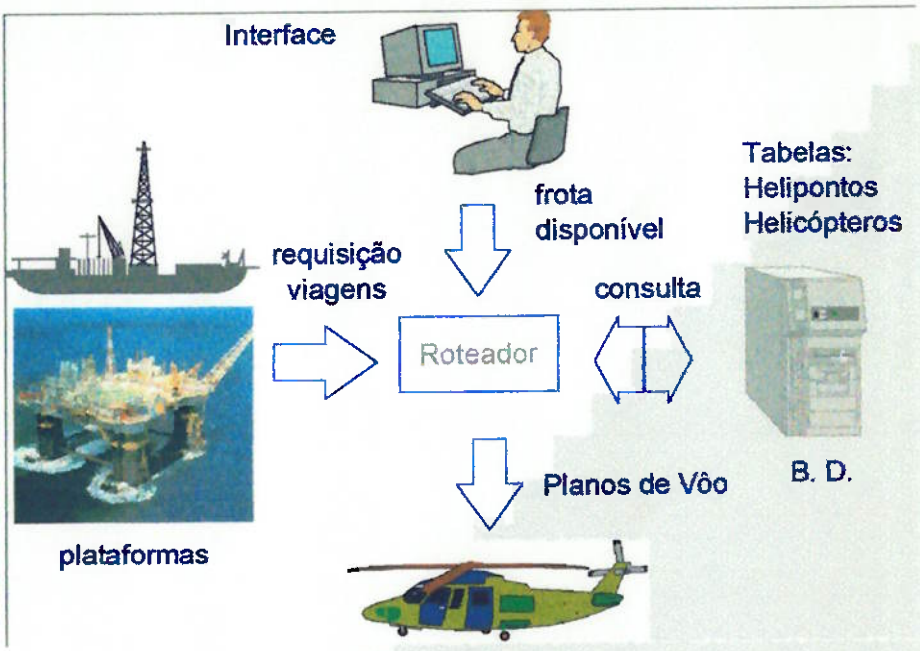
O roteador de helicópteros é constituído de um sistema para elaborar a programação de vôos de helicópteros entre uma base localizada em terra e plataformas de petróleo localizadas no mar. Este tipo de transporte consiste em levar passageiros a partir do aeroporto no litoral, denominado de base, até os heliportos localizados nas plataformas no mar e recolher passageiros a partir destes e levá-los até a base. O objetivo é atender à demanda de passageiros no menor tempo de atendimento e com o menor custo, considerando a disponibilidade de helicópteros.

O sistema roteador é uma aplicação de uma metodologia envolvendo a análise de uma estrutura em Redes de Petri e Inteligência Artificial na solução de um problema de "planning". Nesta abordagem, a partir da análise da estrutura da Rede de Petri do modelo do domínio, pode-se obter critérios de escolha da próxima ação a ser efetuada no plano que está sendo elaborado e, deste modo, dirigir o processo de busca da solução. Desta forma, tem-se um sistema planejador que elabora soluções com mais qualidade. A arquitetura do sistema é de um planejador semi-reativo, ou seja, o sistema sempre tem um plano completo corrente e, além disso, no caso de cancelamento de uma tarefa anteriormente prevista, esta é simplesmente retirada do plano corrente, sem que seja retomado o processo de "planning".

O trabalho de desenvolvimento do software do roteador está previsto para continuar durante o próximo ano. Já foi concluída a primeira etapa, que consiste no estudo do problema, das técnicas utilizadas para a modelagem e do método de solução. Paralelamente a este estudo também foi realizado um treinamento no ambiente de programação Visual PROLOG, no qual o software foi programado.

2. O Roteador de Helicópteros

O sistema Roteador de Helicópteros objeto deste trabalho foi originalmente desenvolvido como aplicação na Tese de Doutorado de Lúcio Mitio Shimada, apresentada à Escola Politécnica da Universidade de São Paulo em 1997. A análise deste sistema tem como objetivo o futuro aperfeiçoamento do software através da complementação de rotinas visando obter melhorias funcionais de ordem prática, no que tange ao funcionamento do programa e à manipulação dos dados, e estética, a partir do desenvolvimento de uma interface gráfica mais completa e amigável.

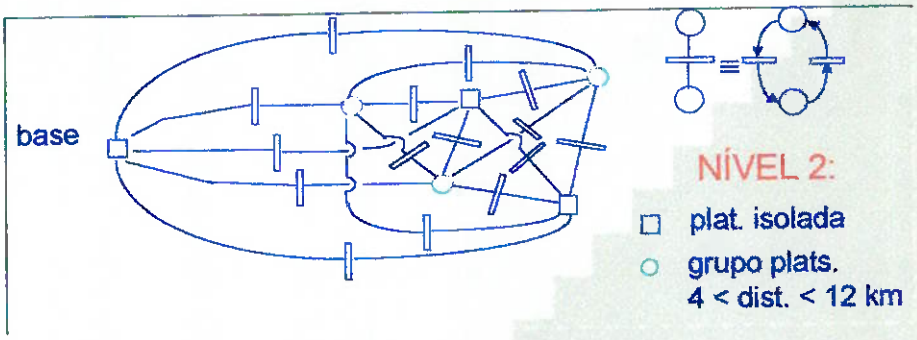


O Roteador de Helicópteros consiste em um sistema para elaborar a programação de vôos de helicópteros entre uma base localizada em terra ("on-shore") e plataformas de petróleo localizadas no mar ("off-shore").

Este tipo de transporte consiste em levar passageiros a partir do aeroporto no litoral, denominado de base, até os helipontos localizadas nas plataformas no mar e recolher passageiros a partir destes e levá-los até a base.

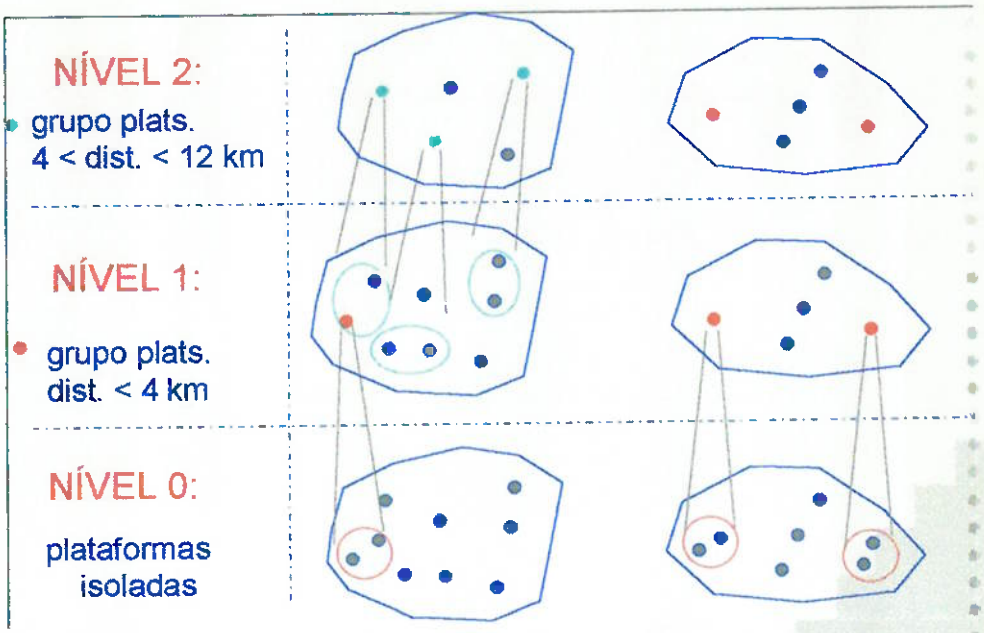
Este sistema é utilizado na rotina do dia-a-dia pelo programador de vôos. Para cada horário de partida do vôo a partir da base, existe uma demanda de passageiros de ida para as plataformas e de volta para a base. O objetivo é atender a demanda no menor tempo de atendimento e com o menor custo, considerando a disponibilidade de helicópteros. Para se ter uma idéia da sua complexidade, este problema envolve 50 helipontos, 16 helicópteros que podem ser de sete tipos diferentes, cinco horários de partida dos vôos e até 20 mil passageiros transportados no mês.

Uma abordagem que associa uma análise da estrutura em Rede de Petri e IA pode melhorar o desempenho na solução do problema geral de "planning". Nesta abordagem, o primeiro passo no desenvolvimento de um sistema de "planning" baseado em IA consiste na síntese de um modelo que representa o domínio do problema baseado em Rede de Petri. A partir da análise da estrutura da RdP deste modelo do domínio, podemos obter critérios de escolha da próxima ação a ser efetuada no plano que está sendo elaborado e, deste modo, dirigir o processo de busca da solução. Desta forma, tem-se um sistema planejador que elabora soluções com mais qualidade, evitando a Anomalia de Sussmann.

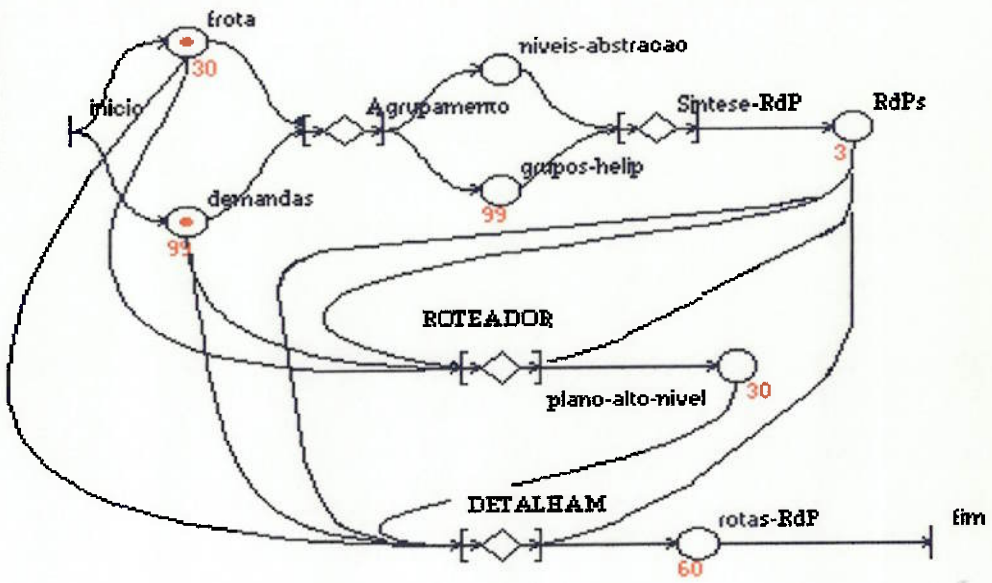


O modelo conceitual do problema foi elaborado usando-se diversos níveis de abstração. Para cada nível de abstração, foram definidos modelos do domínio correspondentes. Nos modelos de alto nível, os lugares são constituídos por grupos de helipontos com coordenadas geográficas de localização próximas. O emprego destes níveis de abstração foi muito útil porque podemos rotear passageiros para ida ou de volta de um grupos de

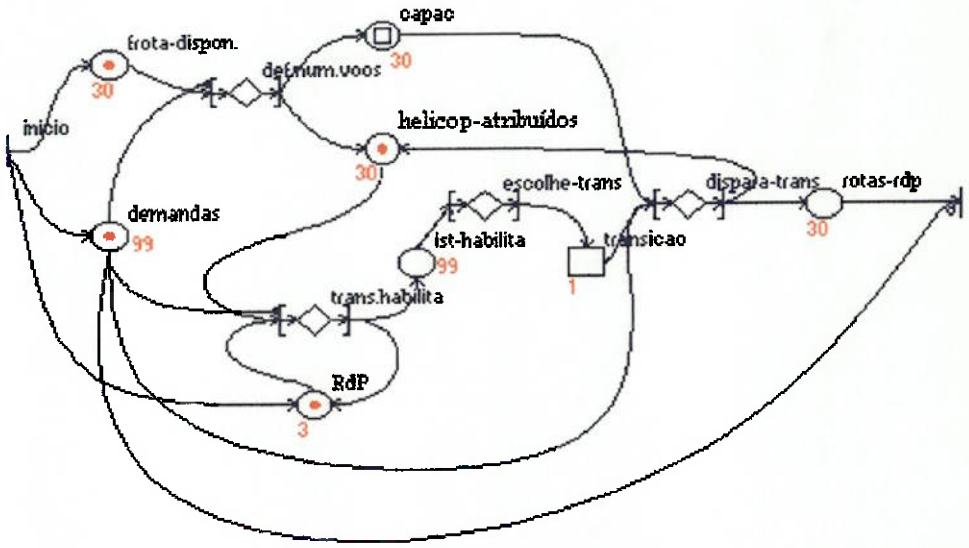
Os níveis de abstração usados foram três: Nível 0, Nível 1 e Nível 2. No Nível 0 ou nível "ground" os boxes da RdP são constituídos pelas plataformas tratadas isoladamente (•). No Nível 1, os boxes representam plataformas isoladas (•) e grupos de plataformas distantes entre si até 4 km (•). No Nível 2, os boxes representam plataformas isoladas (•), grupos de plataformas distantes entre si até 4 km (•) e grupos entre 4 km e 12 km (•).



A figura a seguir mostra a representação usando Rede de Petri do processo de agrupamento de helipontos e de síntese do modelo do domínio. A síntese do modelo do domínio envolve ainda o cálculo das distâncias que correspondem à cada transição. Estas distâncias permitem ao Roteador fazer o cálculo do tempo de vôo estimado e, em conseqüência, o custo estimado da inserção do heliponto correspondente na rota.

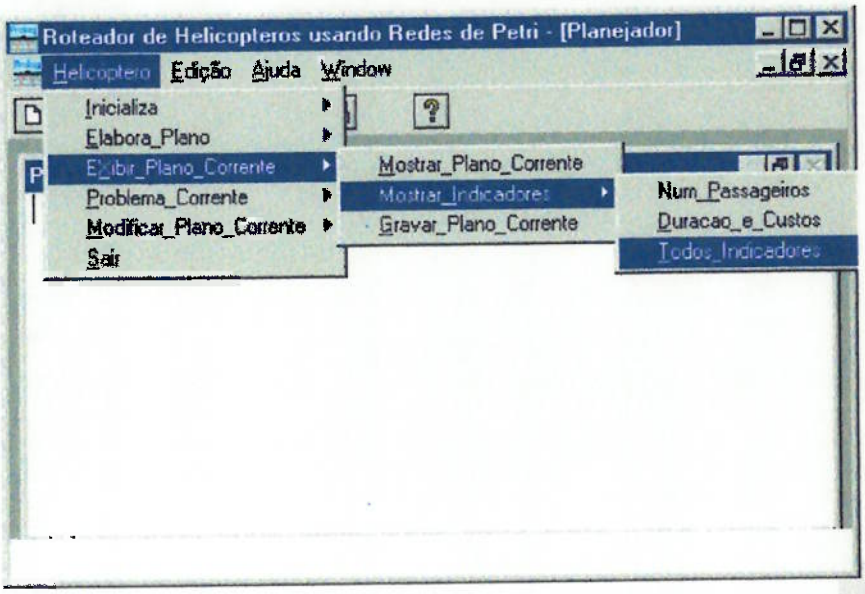


A figura a seguir ilustra uma representação do Roteador, usando uma Rede de Petri. O plano é elaborado incrementalmente pelo Construtor de Planos ou Roteador. O Construtor de Planos é um Ambiente de Rede de Petri que opera sobre a rede que representa o conhecimento do domínio - o "application model".



3. O Software

O software Roteador de Helicópteros foi desenvolvido em Visual PROLOG versão 4.0. Uma ilustração do programa em execução é mostrada na figura a seguir.



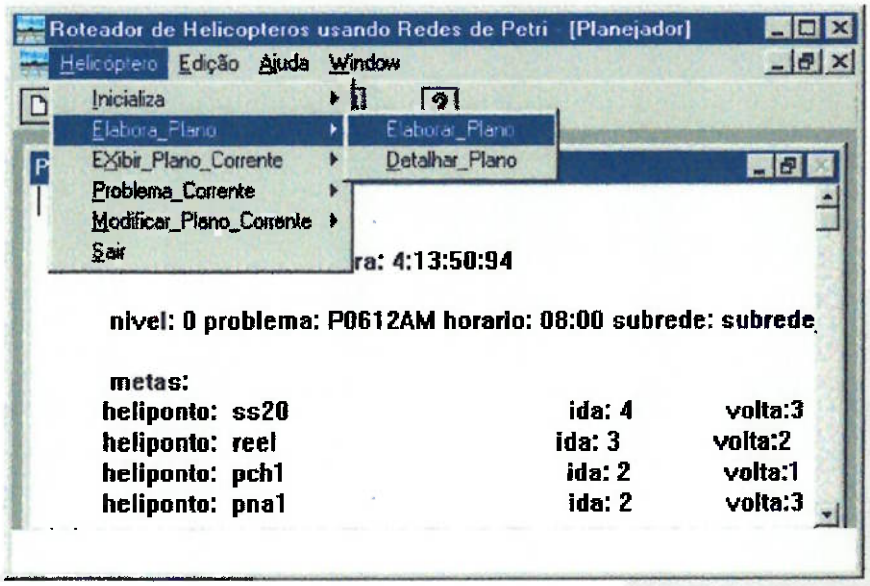
Os dados de entrada são referentes aos helipontos, helicópteros e às requisições de viagens. As informações dos helipontos e helicópteros são fixas e são automaticamente lidas no início da execução do programa. Alternativamente, estes dados podem ser carregados a partir dos menus **Helicóptero | Inicializa | Helipontos | Inicializa_todos** e **Helicóptero | Inicializa | Helicopteros | Inicializa_todos**. Alguns exemplos de dados relativos a helipontos e helicópteros são mostrados a seguir.

```
subrede_a:
HELIPONTO ORDENADA  ABCISSA

      mcae  215308.92  7526587.82
      ss19  313432  7482033
      ss01  314151  7481909
```

```
subrede_a:
HELIPONTO ORDENADA  ABCISSA

          mcae  215308.92  7526587.82
          ss19  313432  7482033
          ss01  314151  7481909
```

Os dados das requisições de viagens (demanda) são lidos a partir de arquivos escolhidos pelo usuário através do menu **Helicóptero | Inicializa | Requisição_Viagens | ler_caso_arquivo**. Estes dados podem ser modificados através da atualização destes arquivos.

O plano de roteamento pode então ser elaborado a partir do menu **Helicóptero | Elabora_Plano | Elaborar_Plano**, como mostrado a seguir:



Um exemplo de um plano obtido a partir do software é listado a seguir. O plano e seus detalhes e parâmetros podem ser visualizados a partir do menu **Helicóptero | Exibir_Plano_Corrente**.

```
Data: 5/11/1998      Hora: 4:13:50:94

nivel:  0  problema:  P0612AM  horario:  08:00  subrede:
subrede_b

metas:
heliponto:  ss20          ida: 4          volta:3
heliponto:  reel          ida: 3          volta:2
```

```
heliponto:  pch1          ida: 2          volta:1
heliponto:  pna1          ida: 2          volta:3
heliponto:  ppg1          ida: 2          volta:1
heliponto:  pvm2          ida: 1          volta:3

helicoptero: b-212 capac.:11
     custo (U$/hr):   374.00  veloc.(km/hr):     185.2
nivel: 0  rota: 1          tempo de voo:   dista(km):
08:00:00  mcae-pch1          00:42:54       132.43
08:42:54  pch1-pna1          00:01:52         5.78
08:44:46  pna1-mcae          00:44:46       138.21
                          Tempo total de Voo:01:29:33
                                 custo (U$):    558.21
                            distancia:     276.419  km

helicoptero: b-212 capac.:11
     custo (U$/hr):   374.00  veloc.(km/hr):     185.2
nivel: 0  rota: 2          tempo de voo:   dista(km):
08:00:00  mcae-ppg1          00:47:56       147.99
08:47:56  ppg1-pvm2          00:03:02         9.39
08:50:59  pvm2-ss20          00:09:24        29.03
09:00:23  ss20-reel          00:06:47        20.94
09:07:10  reel-mcae          00:49:49       153.82
                          Tempo total de Voo:01:57:00
                                 custo (U$):    729.36
                            distancia:     361.169  km

                       custo total (U$):   1287.57
                       distancia total:   637.588   km
                       n.passag.:     27
                 custo unitario (U$/passag.):     47.69
```

Também é possível trocar um helicóptero, eliminar ou incluir um heliponto, depois de já elaborado o plano, sem que seja necessário refazê-lo completamente, através dos comandos do menu **Helicóptero | Modificar_Plano_Corrente**.

3. Modificações

A continuação deste trabalho prevê a implementação das seguintes modificações no software:

1) Desenvolvimento de uma interface gráfica mais amigável, utilizando os recursos disponíveis da programação em Visual PROLOG
2) Implementação de tópicos de ajuda para o software (menu **Ajuda**)
3) Roteamento dos helicópteros considerando a variação da capacidade dos mesmos em função da distância a ser percorrida e do número de pousos e decolagens a ser realizado (devido às diferenças de consumo de combustível)
4) Leitura dos dados de demanda de passageiros diretamente de um banco de dados em rede, e armazenamento dos planos elaborados na forma de relatório
5) Elaboração de um plano para um dia inteiro considerando-se todos os horários programados e a frota disponível em cada horário
6) Modificação do procedimento de alocação dos helicópteros visando otimização: implementação de alocação conforme a demanda, em lugar da pré-alocação por região feita pelo algoritmo atual

16 Bibliografia

[1] SHIMADA, L.M. **Estruturação do Problema de Planejamento em uma Abordagem Baseada em IA e no Formalismo de Redes de Petri.** São Paulo, 1997.

[2] FIKES, R.E; NILSSON, N. **STRIPS: A new approach to the application of theorem proving to problem solving.** 1971. In: ALLEN, J.; HENDLER, J.; TATE, A., coord. **Readings in planning.** San Mateo, California. Morgan Kaufmann, 1990. Cap.2, p.88-97.

[3] SUSSMAN, G.J. **The virtuous nature of bugs.** MIT AI Lab, 1974. In: ALLEN, J.; HENDLER, J.; TATE, A., coord. **Readings in planning.** San Mateo, California. Morgan Kaufmann, 1990. Cap.3, p.111-117.

[4] JACOTO, A. **Programação em PROLOG aplicada a casos de planning em robótica.** São Paulo, USP, 1996. In: Simpósio de Iniciação Científica da Universidade de São Paulo, 4 São Carlos 1996. **Simpósio de Iniciação Científica da Universidade de São Paulo, 4. [Resumos].** São Paulo, USP, 1996. Vol.2, p.336.

[5] JACOTO, A. **Aplicação de Redes de Petri a um problema de planejamento.** São Paulo, USP, 1997. In: Simpósio de Iniciação Científica da Universidade de São Paulo, 5 1997 São Paulo. **Simpósio de Iniciação Científica da Universidade de São Paulo, 5. Resumos.** São Paulo, USP, 1997. Vol.2, p.280.

[6] SANTOS, A.B.; JACOTO, A. **Implementação em Visual Prolog de um roteador de helicópteros.** São Paulo, USP, 1998. In: Simpósio de Iniciação Científica da Universidade de São Paulo, 6 1998 São Carlos. **Simpósio de Iniciação Científica da Universidade de São Paulo, 6. 6.SICUSP : resumos.** São Paulo, USP, 1998. Vol.2, p.359.

[7] SANTOS, A.B.; JACOTO, A. **Implementação em Visual Prolog de um roteador de helicópteros.** São Paulo, USP, 1999. In: Simpósio de Iniciação Científica da Universidade de São Paulo, 7 1999 São Paulo. **Simpósio de Iniciação Científica da Universidade de São Paulo, 7. 7.SICUSP : resumos.** São Paulo, USP, 1999.

[8] PETERSON, J.L. **Petri Net Theory and the Modeling of Systems.** Englewood Cliffs, Prentice-Hall, 1981.

[9] NILSSON, N.J. **Principles of Artificial Intelligence.** S.L., Morgan Kaufmann Publishers, 1993.

[10] CHAKRABARTY, S., WOLTER, J.A. **Structure-oriented Approach to Assembly Sequence Planning.** IEEE, 1997.

[11] CLOCKSIN, W.F.; MELLISH, C.S. **Programming in Prolog.** Berlim, Springer-Verlag, 1984.

[12] ARITY CORPORATION. **The Arity/PROLOG Language Reference Manual.** Massachusetts, 1988.

[13] PROLOG DEVELOPMENT CENTER A/S. **Visual PROLOG Version 4.0 - Language Tutorial.** Denmark, 1996.

[14] PROLOG DEVELOPMENT CENTER A/S. **Visual PROLOG Version 5.0 - Language Tutorial.** Denmark, 1997.