LÚCIO MITIO SHIMADA

Volume 2 / 2

ESTRUTURAÇÃO DO PROBLEMA DE PLANEJAMENTO EM UMA ABORDAGEM BASEADA EM IA E NO FORMALISMO DE REDES DE PETRI

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia.

São Paulo

1997

OK

LÚCIO MITIO SHIMADA

Volume 2 / 2

APÊNDICES

ESTRUTURAÇÃO DO PROBLEMA DE PLANEJAMENTO EM UMA ABORDAGEM BASEADA EM IA E NO FORMALISMO DE REDES DE PETRI

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia.

Área de Concentração:
Engenharia Mecânica

Orientador:
Prof. Dr. José Reinaldo Silva

São Paulo

1997

Universidade de São Paulo - Escola Politécnica (EPUSP)
Faculdade de Engenharia Mecânica
Laboratório de Automação de Sistemas (LAS) - *Eng. Mecatrônica*

Aluno: Lúcio Mitio Shimada N.USP: 1681108
Data: 02 de abril de 1997.

Assunto: Tese de Doutorado em
Engenharia Mecatrônica

Título: Estruturação do Problema de Planejamento em uma Abordagem Baseada em IA e no Formalismo de Redes de Petri

Palavras-chaves: Redes de Petri, PFS / MFG, Ghenesys, HIPER, Inteligência Artificial, PROLOG, Planejamento e Sequenciação, Sistemas Dinâmicos, Roteador de Helicópteros, Planejamento Semi-Reativo, Problemas Combinatórios, Metodologia de Desenvolvimento de Sistemas, Estruturação, Modelo Conceitual.

AGRADECIMENTOS

Ao meu amigo Prof. Dr. José Reinaldo Silva, pela orientação deste trabalho. Ao Prof. Dr. Márcio Rillo, pela orientação nos estudos em AI Planning. Ao Prof. Dr. Paulo Eiji Miyagi, pelos estudos em PFS / MFG. Aos meus colegas da Mecatrônica, Prof. José Ricardo Sebastião, Prof. Diolino dos Santos Fo., Prof.a. Cristina Toshie Motohashi, Prof. Júlio Arakaki. À todos que direta ou indiretamente, contribuíram na execução deste trabalho. E, finalmente, à CNPq, FAPESP, FlexSys (Esprit 76101), AP3I e PETROBRÁS, especialmente ao Eng. Nelson Costa Cardoso, Chefe da Divisão Regional de Informática em São Paulo - DIRINF-SP, pelo patrocínio para a realização deste trabalho.

Aos meus pais,

Hiroshi e

Margarida Shimada.

SUMÁRIO

APÊNDICES

APÊNDICE I

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

I PLANEJAMENTO E SEQÜENCIAÇÃO

O planejamento num sistema de manufatura discreta, consiste na definição das ações necessárias para se atingir determinados objetivos no contexto do sistema organizado. No conceito de Sistema Integrado de Manufatura (SIM), esta função corresponde ao Sistema de Planejamento e Programação (PS) de um Sistema Flexível de Fabricação (FF). A solução para o problema do planejamento é obtida tradicionalmente com o emprego da programação matemática. Esta constrói modelos que correspondem aos diferentes elementos que compõe a empresa. O objetivo de um modelo é a de maximizar o retorno financeiro ou a de minimizar as perdas no processo obedecidas as limitações na capacidade dos recursos produtivos (equipamentos, pessoal, insumos, capital) e questões tecnológicas, tais como: qualidade de produto, segurança industrial e meio ambiente. O resultado do modelo apoia a decisão acerca da definição das quantidades agregadas de cada tipo de insumo a ser utilizado e dos produtos a serem produzidos para um determinado intervalo de tempo, por exemplo, um mês.

A seqüenciação de operações num SFF é um problema da função Planejamento do Processo Auxiliado por Computador - PPAC. As metas agregadas requerem a execução de tarefas que, por sua vez, implicam na execução de diversas operações. Os roteiros de produção para cada tipo de peça indicam a ordem parcial entre operações, os requisitos de máquinas e a duração de cada operação. A seqüenciação consiste na escolha da seqüência de operações (dentro do roteiro de produção), na alocação de recursos à estas e na definição do horário do seu início. Na seqüenciação, um conjunto de peças deve ser processado em máquinas de modo a minimizar (ou maximizar) determinadas medidas de desempenho. Uma tarefa (ou ordem de fabricação de peças) pode compreender diversas operações a serem efetuadas em diversas máquinas. Nenhuma máquina pode operar mais de uma peça ao mesmo tempo e nenhuma peça pode ser trabalhada em mais de uma máquina ao mesmo tempo.

Existem basicamente dois modos de seqüenciação, dependendo da maneira como as peças visitam as máquinas: seqüenciação por fluxos ("flow shop") ou seqüenciação por tarefas ("job shop").

No modo "flow shop" todas as peças fluem numa mesma direção e cada máquina é visitada uma única vez. No modo "job shop" as peças podem fluir em direções diferentes e cada máquina pode ser visitada mais de uma vez por uma mesma peça. Portanto, o modo "job shop" é o modo mais flexível de seqüenciação que existe. Em ambos os modos não é necessário que uma peça visite todas as máquinas da Célula de Fabricação.

Uma questão importante é a diferença entre os problemas de planejamento e de seqüenciação. [Fox 1984] classifica a escolha do processo produtivo (ou roteiro de produção) como sendo uma tarefa de planejamento, e a alocação de recursos e a definição do horário do início das operações como sendo tarefas de seqüenciação. Em outras palavras, o planejamento define o que (e quanto) deve ser feito, enquanto a seqüenciação define quando o que foi planejado deve ser feito. A forma como este é feito determina o grau de dependência entre estes dois problemas. Estes problemas podem ser desacoplados e resolvidos separadamente apenas quando a dependência entre elas é fraca. Se a dependência é forte, a forma como a execução do plano é executado impõe limitações ao que (e a quanto) se pode fazer e a quando fazer. Neste caso, [Fox 1984] recomenda que estes sejam resolvidos simultaneamente quando existe uma dependência forte. Por exemplo, a escolha correta de um roteiro de produção (planejamento) requer que se faça simultaneamente a alocação de recursos e a definição do horário do início das operações (seqüenciação). A factibilidade de um roteiro de produção depende da factibilidade de cada uma das operações incluídas no roteiro e, além disso, uma determinada operação só está factível se os seus requisitos de recursos estiverem satisfeitos na hora em que a operação deve ser feita.

Em relação à complexidade do problema da seqüenciação, evidentemente, podem existir infinitas seqüenciações factíveis para se atingir um mesmo objetivo agregado, o que caracteriza este problema como sendo do tipo NP-completo. Por exemplo, a seqüenciação de apenas 10 tarefas consistindo de cinco operações onde associamos uma

única máquina à cada operação (i.e., não há roteiros de produção alternativos) apresenta um espaço de busca com $(10!)^5$ ou cerca de 10^{32} seqüenciações possíveis. Por este motivo, a abordagem da IA surge como uma alternativa atraente para a resolução deste tipo de problema.

O problema do planejamento e da seqüenciação têm uma natureza combinatória, sendo bem conhecida a complexidade dos algoritmos exatos para a resolução destes problemas. As técnicas usadas na IA podem obter soluções boas, embora não ótimas, num tempo de processamento comparativamente bem menor que estes algoritmos exatos. A questão está na escolha de critérios (ou "planning rationale") para a tomada de decisão judiciosa durante a síntese de uma solução, como mostramos em nossa proposta de metodologia descrita no capítulo 4. Outros trabalhos anteriores já provaram a expressividade da rede de Petri para representar problemas de scheduling. [Valette et Silva 1989] descrevem o processo de representação de um problema de scheduling job shop em rede de Petri temporizada do tipo C / E, como mostra a Figura I.1, a seguir.

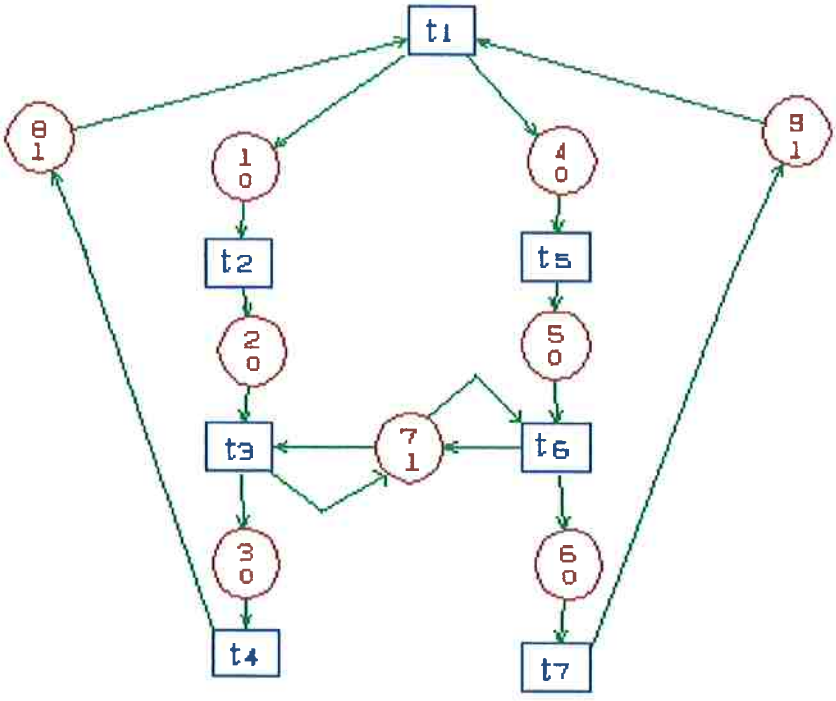


Figura I.1 - Scheduling usando uma RdP C / E temporizada

Os lugares da RdP são: 1-9. As transições da RdP são: *t1-t7*. As transições *t3* e *t6* estão em conflito porque disputam o recurso compartilhado 7. Os tempos associados com as transições temporizadas estão na Tabela I.1, a seguir. O autor ilustra a natureza não linear do problema de scheduling, observando que, usando a regra de despacho "disparar a transição mais cedo possível", obtém-se uma duração total ("makespan") de 8 horas. Esta solução está na Figura I.2.a, a seguir. Observe que o caminho crítico nesta solução passa pelo uso do recurso compartilhado 7.

Tabela I.1 - Durações associadas com as transições da RdP na Fig. I.1

Transição	Duração	Pré-Condições	Pós-Condições
t1	1	8, 9	1, 4
t2	1	1	2
t3	2	2, 7	3, 7
t4	1	3	8
t5	2	4	5
t6	1	5, 7	6, 7
t7	3	6	9

Paradoxalmente, o makespan pode ser reduzido de 8 para 7 horas, se aumentarmos a duração de uma das tarefas, no caso a tarefa *t2*, de 1 para 3 horas. A solução para o problema após esta modificação está na Figura I.2.b. Podemos concluir, a partir deste resultado, que a teoria da Rede de Petri não é capaz de distinguir as boas soluções para o problema de Scheduling. Por este motivo, na nossa proposta, descrita no capítulo 4, propomos a utilização de Rede de Petri associado com a Inteligência Artificial (IA).

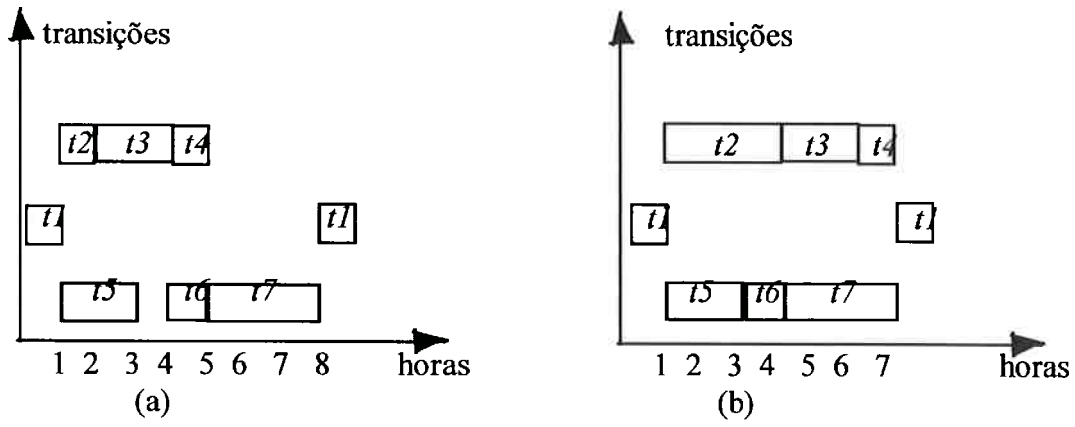


Figura I.2 - Solução para o problema de Scheduling da RdP na Figura I.1

A seguir, mostramos a aplicação da metodologia proposta que fizemos para Sistemas Flexíveis de Fabricação. Esta aplicação foi escrita em Visual PROLOG para Windows 3.11.

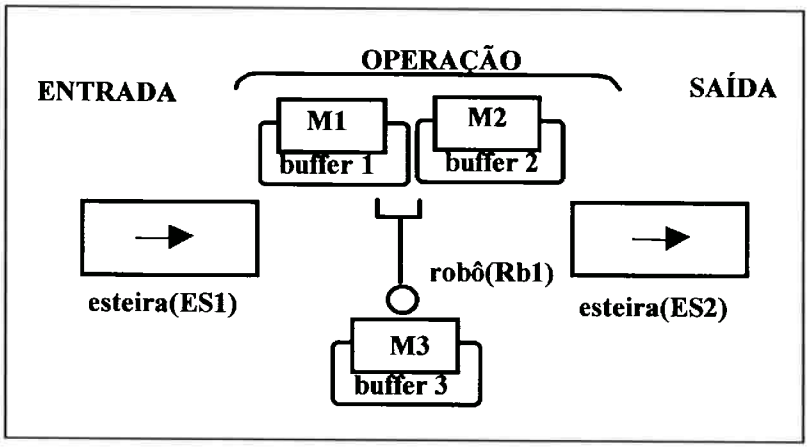


Figura I.3 - Exemplo de um SFF

I.1 - Um Problema de Scheduling Job Shop

Vamos apresentar agora, uma aplicação da metodologia proposta no capítulo 4 para o problema de scheduling Job Shop numa Célula de Manufatura Flexível. Seja uma Célula Flexível composta por três máquinas, um robô e duas esteiras de transporte, como ilustra a Figura I.3, acima. Inicialmente, vamos mostrar o expressividade da Rede de Petri temporizada usando um exemplo de [Miyagi 1990], ilustrado na Tabela I.2 seguinte.

Tabela I.2 Descrição das Operações Necessárias e do Uso de Recursos

Peça	P1	P2	P3	P4
Operação	O1 O2 O3	O4 O5 O6	O7 O8 O9	O10 O11 O12
Máquina	M1 M2 M3	M3 M1 M2	M1 M3 M2	M2 M3 M1
Tempo	5 8 2	7 3 9	1 7 10	4 11 7

Problema: Seqüenciação "job shop" de quatro peças (ou ordens de serviço) utilizando uma Célula Flexível com três máquinas está descrito na Tabela I.2, acima. O valor Mk(t) indica que a operação Opi (i=1,2,3) deve ser feita na máquina Mk (k=1,2,3) e esta operação requer um tempo t de processamento.

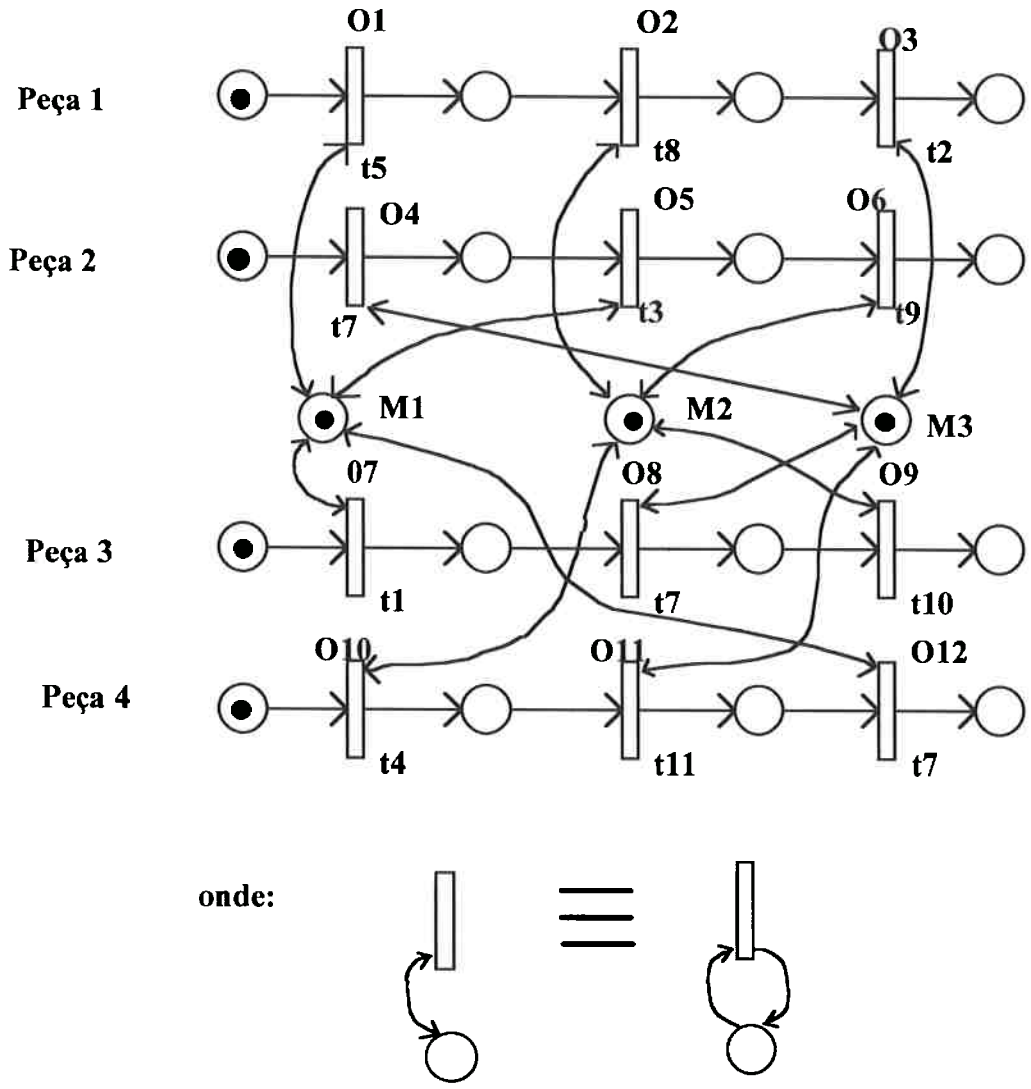


Figura I.4 - "Application model" para o problema da Tab. I.2 (RdP temporizada)

A Figura I.4, acima, ilustra uma rede de Petri C / E com transições temporizadas, que representa o problema descrito na Tabela I.2, acima [Miyagi 90]. Esta rede representa o domínio do problema, i.e., o "application model", enfatizando aspectos tecnológicos como a modelagem dos roteiros de produção das peças. A grande vantagem deste tipo de representação está na sua capacidade de capturar a totalidade das soluções de um problema. Entretanto, ela não pode indicar nenhuma solução em particular e nem como o plano deve ser elaborado. Este conhecimento envolve a estratégia de resolução do problema (e é distinto portanto do conhecimento do domínio da aplicação). Para tanto, é necessário desenvolver um novo modelo de representação -- o "planning self model".

A Figura I.5, seguinte, ilustra a rede de Petri estendida Ghenesys que representa este problema de scheduling em si, ou seja o "planning self model".

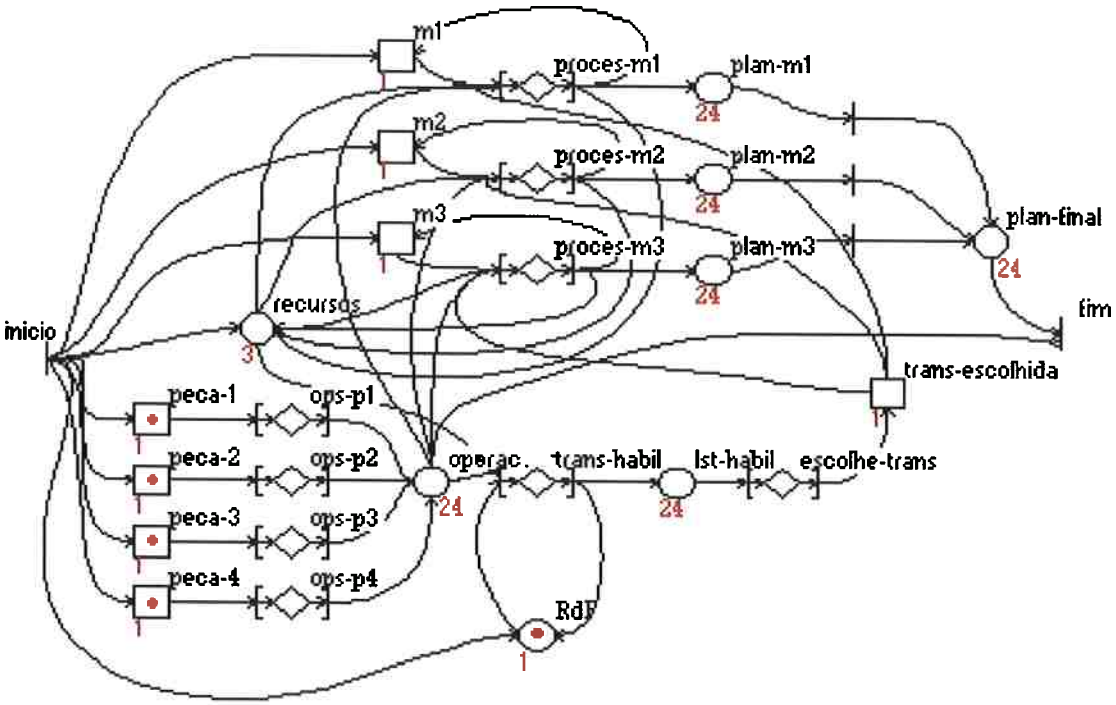


Figura I.5 - "Planning self model" para o scheduling Job Shop

Na Figura I.5, os boxes peça-1, peça-2, peça-3 e peça-4 representam demandas ou ordens de fabricação. Havendo uma marca nestes boxes, faz-se o desdobramento destas ordens de fabricação em operações a serem escalonadas. Os boxes m1, m2 e m3 representam recursos compartilhados. Um determinado roteiro de produção será factível (ou não), dependendo da disponibilidade (ou não) destes recursos. O número de recursos disponíveis é indicado pelo box recursos. Existindo pelo menos uma marca no box recursos, faz-se a verificação das transições habilitadas. Se o box capacidade que representa a lista de transições habilitadas tiver marcas, faz-se a escolha da transição a ser disparada. O disparo da transição eqüivale ao início de uma operação usando um determinado recurso. Enquanto a operação estiver sendo feita, esta prende o recurso durante um determinado tempo de duração. Nesse intervalo, havendo outros recursos disponíveis, repete-se o procedimento acima. Caso contrário, é necessário esperar a liberação de algum recurso ao término de alguma operação para reiniciar um novo ciclo de alocação de

tarefas. Quando a lista de operações pendentes estiver vazia, tem-se o plano final para o problema de scheduling.

Este exemplo foi implementado em Visual PROLOG. Apesar de muito singelo, mostramos que é possível se fazer uma aplicação da metodologia proposta no capítulo 4 para os Sistemas Flexíveis de Fabricação (SFF). No desenvolvimento do sistema de scheduling em Visual PROLOG, faz-se a implementação do modelo conceitual (*mc*), ou seja, a implementação do "planning model" e do "application model".

Uma possível critica pode ser feita com respeito ao tipo de rede de Petri usada para representar o domínio da aplicação ("application model"). Evidentemente, para problemas reais em FMS, uma representação em RdP do tipo C/E apresenta o inconveniente de crescer exponencialmente. Devemos esperar que problemas reais resultem em grafos muito grandes onde a complexidade de uma análise computacional cresce exponencialmente. Mesmo em sistemas flexíveis para pequena e média empresa, e portanto menores, uma análise estrutural é proibitiva. Portanto procuramos utilizar redes mais sintéticas como é o caso do Ghenesys que introduziremos a seguir. Em nossa aplicação em Visual PROLOG, resolvemos ambos os problemas: o que está descrito acima e o que descrevemos em seguida.

Tabela I.3 Roteiros de Produção Existentes para Manufatura num SFF

Peça		P1	P2	P3	P4
Tarefas		O1 O2 O3	O4 O5 O6	O7 O8 O9 O10	O11 O12
Pre-C		- O1 O2	- - O4,O5	- - - O9	- -
Proces Básico	Máq.	M2 M1 M3	M1 M2 M3	M3 M1 M3 M1	M3 M2
	Tempo	4 5 2	6 3 3	3 3 6 2	4 3
Proces Altern.	Máq.	M3 M2 M1	M2 M3 M1	M3 M3 M2 M3	M2 M1
	Tempo	6 6 4	6 4 5	4 5 7 2	4 5

I.2 - Outro Problema de Scheduling Job Shop

[Silva et Shimad 95], apresentam uma aplicação mais realística das idéias centrais relacionadas com este trabalho. Também um problema de seqüenciação de tarefas num SFF do tipo "job shop" com quatro tipo de peças e com três máquinas, como ilustrado na Figura I.3. Cada peça apresenta dois roteiros de produção (um processo básico e um processo alternativo); como mostra a Tabela I.3, na página anterior.

A elaboração da rede de Petri Ghenesys que representa o domínio da aplicação do problema ("application domain model") deve ser elaborada, de acordo com os seguintes passos da metodologia descrita no capítulo 4, item 4.5.1.

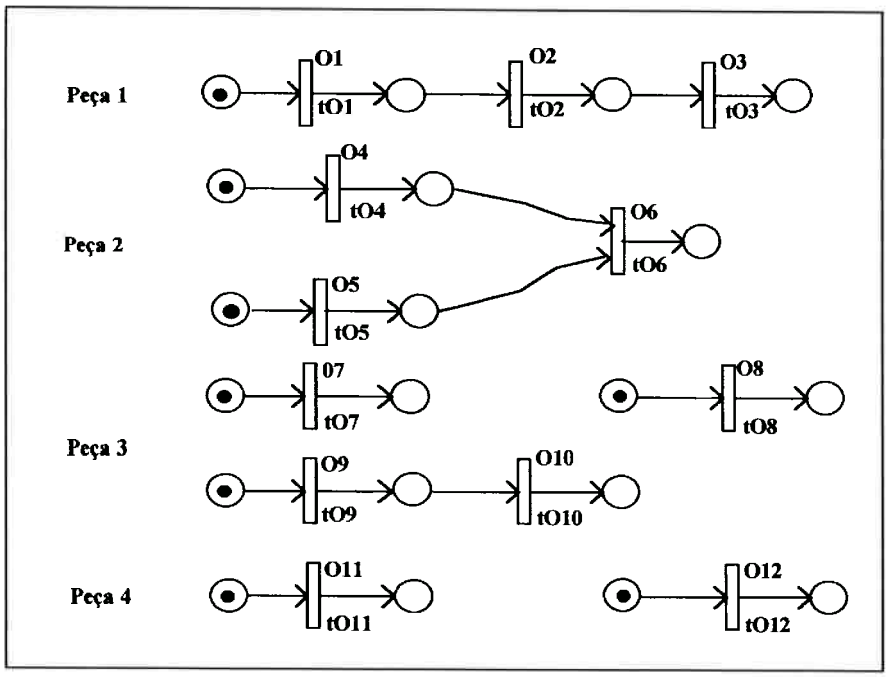


Figura I.6 - Rede C/E de Ordem parcial das Operações em cada Peça

PASSO1: Vamos considerar o Sistema Flexível de Fabricação como um sistema por Eventos Discretos. Os elementos fundamentais para o planejamento do sistema dinâmico são: horizonte de planejamento (τ), demanda, linha de produtos (P1, P2, P3 e P4), recursos produtivos (M1, M2 e M3), roteiro de produção e factibilidade do plano. Apenas para ilustrar, vamos elaborar uma

rede de Petri do tipo C / E temporizada para representar a ordem parcial das operações para o problema definido na Tabela I.3. A Figura I.6, acima, ilustra esta RdP C/E. Observe que nada foi representado acerca dos recursos compartilhados, nem dos processos de produção básico ou alternativo. Mesmo assim, o tamanho da rede de Petri já é bastante significativa. Por este motivo, deve ser usada a rede mais sintética do tipo Ghenesys para se fazer a representação do domínio do problema.

PASSO2: Elaboração do "application model" para o problema de scheduling Job Shop, usando Rede de Petri do tipo Ghenesys . A Fig. I.7, a seguir, mostra a rede Ghenesys para o sistema produtivo ilustrado na Figura I.3.

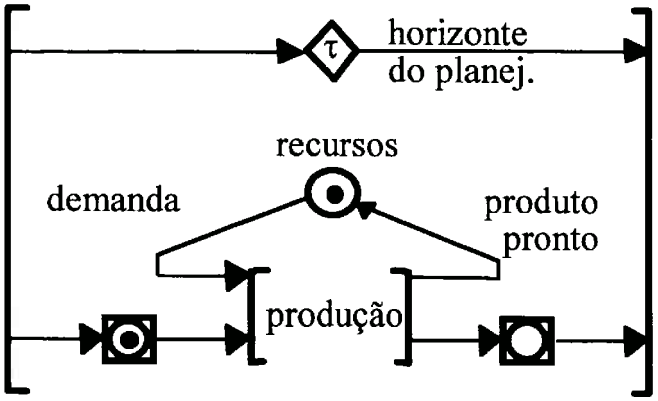


Figura I.7 - Rede Ghenesys para o Scheduling do sistema produtivo na Fig. I.3

Níveis de abstração no Ghenesys :

Descrevemos no capítulo 3 como, o Ghenesys provê, em controle, a integração total do sistema produtivo e o sistema de informações numa única arquitetura hierárquica. A Figura I.8, seguinte, mostra como os diversos níveis de abstração no Ghenesys representam as áreas funcionais do SIM: Controle, PPAC (Programação da Produção Auxiliado por Computador), SPP (Sistema de Planejamento da Produção) e a alta direção da empresa. Observe que o grau de detalhamento aumenta à medida que se caminha no sentido da camada de maior abstração para as camadas de menor nível de abstração. Por este motivo, na elaboração da rede Ghenesys , é recomendada a abordagem "top-down". A Fig. I.7, acima, mostra a rede Ghenesys que repre-

senta o plano de mais alto nível, no nível de gestão ou de SPP. Admite-se o uso da política de estoque "just-in-time", i.e., não há nenhum estoque de insumos ou produtos. Este plano deve ser detalhado para o nível de processo para atender as restrições tecnológicas: roteiros de produção e alocação de recursos compartilhados.

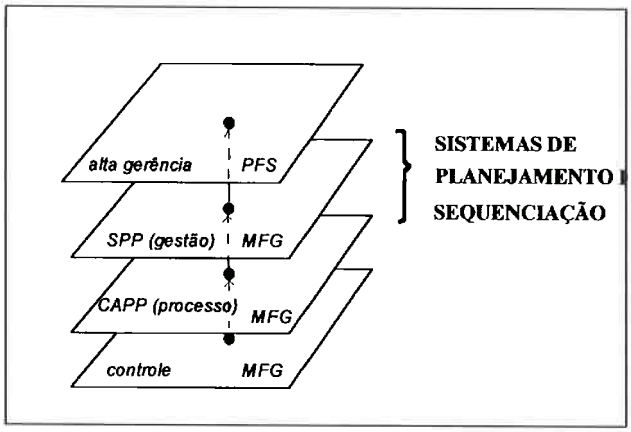


Figura I.8 - Níveis de abstração na rede estendida Ghenesys

Vamos apenas ilustrar este processo de detalhamento da RdP Ghenesys para os diferentes níveis de abstração, entretanto, sem completar o processo todo.

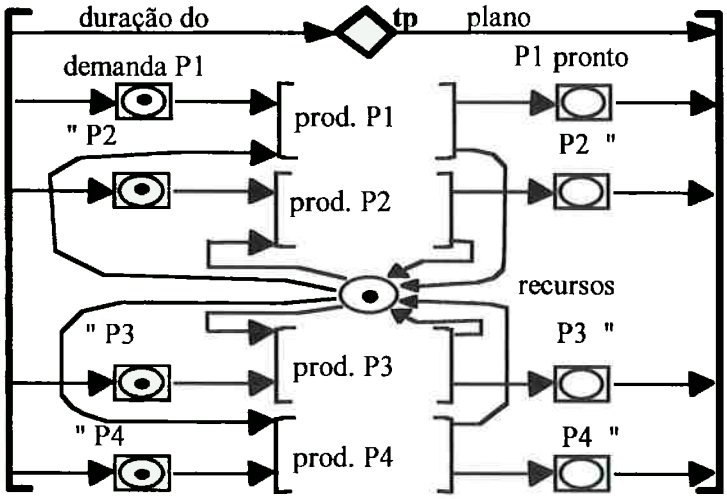


Fig. I.9 - Rede Ghenesys da atividade de produção

A Fig. I.9, acima, mostra a rede Ghenesys para a atividade de produção. A Figura I.10, a seguir, mostra a rede Ghenesys para a atividade de produção da peça P1.

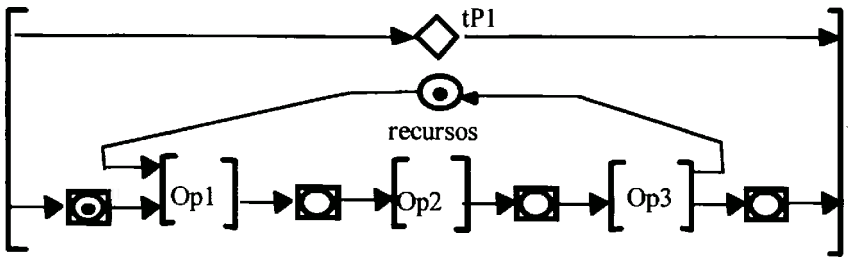


Figura I.10 - Rede Ghenesys da peça P1

A duração plano **tp** = max(tPi, i=1,,4). Para que o plano seja factível, a duração do plano **tp** deve ser menor que o horizonte do planejamento τ. A Fig. I.11.a mostra a rede Ghenesys para os roteiros de produção para a operação Op1 na peça P1.

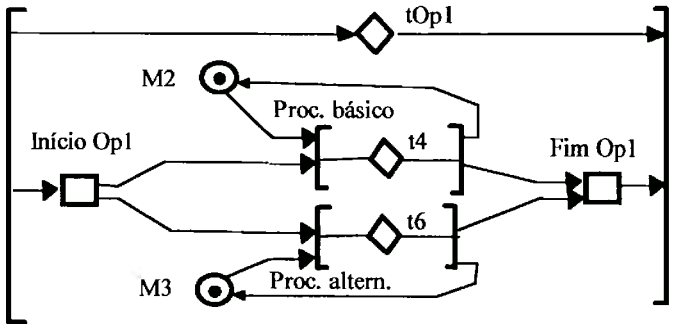


Figura I.11.a - Rede Ghenesys para a operação Op1 na Peça P1

A duração da Op1 depende tOp1 depende do roteiro escolhido. Observe, na Figura I.11.a que a duração da operação tO1 depende do roteiro de produção escolhido para a sua realização, tO1 = 4, se o processo escolhido for básico, e tO1 = 6, se o processo escolhido for alternativo. Deste modo, temos uma estimativa otimista e outra pessimista para tO1, i.e. (4 <= tO1 <= 6). Tem-se, deste modo, um estimador grosseiro para a **tp**. Esta é feita a nível de gestão. Se o horizonte do planejamento τ for maior que **tp** (pessimista e / ou otimista) estimada, o plano poderá ser detalhado no nível mais abaixo.

Em termos de nível de abstração, podemos dizer que as RdPs nas Figuras I.9, I.10 e I.11.a estão localizadas no nível do processo (CAPP) do SIM. Por sua vez, a RdP na Figura I.7 está localizada no nível de gestão (SPP) do SIM.

PASSO3: Análise das RdPs Ghenesys que representam o "application model". Através das Figuras I.7 a I.11.a observamos que existem conflitos no uso de recur-

sos compartilhados. Além disso, através da Fig. I.11.a podemos ver que existem roteiros alternativos de produção para cada operação, com tempos de processamento, em geral, maiores. Estes conflitos são:

a) no processo básico:
   {O2,O4,O8 e O10} compartilham a máquina M1,
   {O1,O5 e O12} compartilham a máquina M2,
   {O3,O6,O7,O9 e O11} compartilham a máquina M3.
b) no processo alternativo:
   {O3,O6 e O12} compartilham a máquina M1,
   {O2,O4,O9 e O11} compartilham a máquina M2,
   {O1,O5,O7,O8 e O10} compartilham a máquina M3.

Estes conflitos não podem ser resolvidos pela RdP. Portanto, vamos temos de regras de prioridade obtidas a partir da teoria da seqüenciação para resolve-los. O sistema é flexível quanto à inclusão de regras para resolução de conflitos. A partir do conjunto de operações seqüenciáveis, pode-se escolher uma determinada operação usando as seguintes regras de prioridade utilizadas na engenharia de produção:

R1: a operação com o menor tempo de processo (ou spt - "shortest processing time")
R2: a operação com o maior número de operações sucessivas
R3: a operação pertencente à peça com o menor número de operações seqüenciáveis
R4: a operação com o maior número de operações imediatamente sucessivas
R5: a operação pertencente à peça com o maior número de operações não processadas
R6: a operação pertencente à peça com o menor tempo de folga

Em nossa implementação, incluímos apenas a regra R1.

PASSO4: Elaboração do "planning self model" para o problema de scheduling Job Shop, levando em conta os resultados obtidos no PASSO3. A configuração da rede de Petri Ghenesys que representa este problema de scheduling Job Shop continua sendo a mesma mostrada na Fig. I.5. A diferença está apenas no fato de que os roteiros de produção são mais complexos, e envolvem processos alternativos e o uso de máquinas substitutas, com alguma penalidade (no caso, consideramos apenas uma duração maior para a tarefa).

PASSO5: implementamos o sistema modelado nos passos PASSO2 e PASSO4 usando uma linguagem de programação simbólica, por exemplo o Visual PROLOG.

PASSO6: os resultados computacionais estão descritos no item I.4, a seguir.

I.3 - A Implementação do Sistema

A implementação do sistema foi feita na linguagem Visual PROLOG para Windows 3.11, em dois módulos separados:

- módulo planejador baseado em Rede de Petri, e
- módulo de interface com o usuário.

O módulo planejador, basicamente, implementa o Planejador baseado em Rede de Petri. O plano é elaborado incrementalmente, usando-se um ambiente de redes de Petri que opera sobre a rede que representa o conhecimento do domínio. Conseqüentemente, todas as ordens de serviço são escalonadas em paralelo. A cada passo, o Planejador elabora uma lista de transições habilitadas. É utilizado um critério_corrente para a escolha da transição. Então, faz-se o disparo da transição, atualizando todos os estados da RdP, passando-se, então, ao passo seguinte.

O exemplo da Tabela I.3 é mais completo, incorporando processos alternativos de produção e permitindo até a escolha de diferentes estratégias para resolução de conflito. A regras de despacho ("dispatching") podem levar em conta as diferentes estratégias da engenharia de produção, tais como: minimizar ociosidade das máquinas, minimizar tem-

po total de processo, minimizar estoque de peças em processamento, etc. Veja [Kusiak 1990] para análise mais detalhada destas estratégias. A prioridade na aplicação da regra de despacho pelo sistema pode ser alterada, por exemplo de alocar primeiro operação com menor tempo de processo ("shortest processing time") para alocar primeiro a operação com maior número de peças subsequentes, etc.

O módulo de interface permite ao usuário definir o problema corrente para o Planejador. A carga do problema corrente é feito através da leitura de termos em PROLOG gravados em arquivo. Os termos definem a RdP correspondente ao problema e a tradução dos elementos da RdP para o domínio do problema.

Como referência para a linguagem PROLOG, veja [Clocksin et Mellish 1987], [Sterling et Shapiro 1986] e [Araribóia 1989]. Para obter referência do Visual PROLOG, veja [PDC 1996].

I.4 - Resultados Computacionais

Vamos mostrar que o módulo Planejador independe dos dados, através da apresentação dos resultados computacionais obtidos. Apresentamos os resultados tanto para o problema definido na Tabela I.2 e como também para o problema definido na Tabela I.3.

I.4.1 - Problema exemplo da Tabela I.2

Inicialmente, vamos considerar o problema definido na Tabela I.2 O Menu do sistema tem a função para *Ler_Arquivo_Processos*. Nesta função, a definição da RdP que descreve o domínio do problema é lida na forma de termos Visual PROLOG gravados em arquivo. O arquivo miyagi01.rdp contem os termos correspondentes à Tab. I.2.

Em seguida, deve-se definir as ordens de fabricação e os recursos disponíveis. Por exemplo: definimos demandas para: p1, p2, p3 e p4 com os recursos: m1, m2 e m3 disponíveis. Ativando a função Solucionar do Menu, temos a solução que está listada

em seguida. Esta solução tem uma duração ("makespan") de 34 horas. O sistema pode apresentar o plano global ou o plano detalhado para cada recurso.
A listagem a seguir, ilustra o plano por recursos:

Listagem do Plano por Recursos: Data: 18/11/1996 Hora: 13:42:54:77

i) Recurso: m1

tempo: 0	inicia: op7	recurso alocado: m1	duração: 1	término: 1
tempo: 1	termina: op7	recurso liberado: m1	início: 0	duração: 1
	inicia: op1	recurso alocado: m1	duração: 5	término: 6
tempo: 6	termina: op1	recurso liberado: m1	início: 1	duração: 5
tempo: 7	inicia: op5	recurso alocado: m1	duração: 3	término: 10
tempo: 10	termina: op5	recurso liberado: m1	início: 7	duração: 3
tempo: 27	inicia: op12	recurso alocado: m1	duração: 7	término: 34
tempo: 34	termina: op12	recurso liberado: m1	início: 27	duração: 7

ii) Recurso: m2

tempo: 0	inicia: op10	recurso alocado: m2	duração: 4	término: 4
tempo: 4	termina: op10	recurso liberado: m2	início: 0	duração: 4
tempo: 6	inicia: op2	recurso alocado: m2	duração: 8	término: 14
tempo: 14	termina: op2	recurso liberado: m2	início: 6	duração: 8
	inicia: op6	recurso alocado: m2	duração: 9	término: 23
tempo: 23	termina: op6	recurso liberado: m2	início: 14	duração: 9
	inicia: op9	recurso alocado: m2	duração: 10	término: 33
tempo: 33	termina: op9	recurso liberado: m2	início: 23	duração: 10

iii) Recurso: m3

tempo: 0	inicia: op4	recurso alocado: m3	duração: 7	término: 7
tempo: 7	termina: op4	recurso liberado: m3	início: 0	duração: 7
	inicia: op8	recurso alocado: m3	duração: 7	término: 14
tempo: 14	termina: op8	recurso liberado: m3	início: 7	duração: 7
	inicia: op3	recurso alocado: m3	duração: 2	término: 16
tempo: 16	termina: op3	recurso liberado: m3	início: 14	duração: 2
	inicia: op11	recurso alocado: m3	duração: 11	término: 27
tempo: 27	termina: op11	recurso liberado: m3	início: 16	duração: 11

A Figura I.12, seguinte, ilustra o plano listado acima para o problema definido na Tabela I.2 e elaborado pelo Planejador baseado em Rede de Petri.

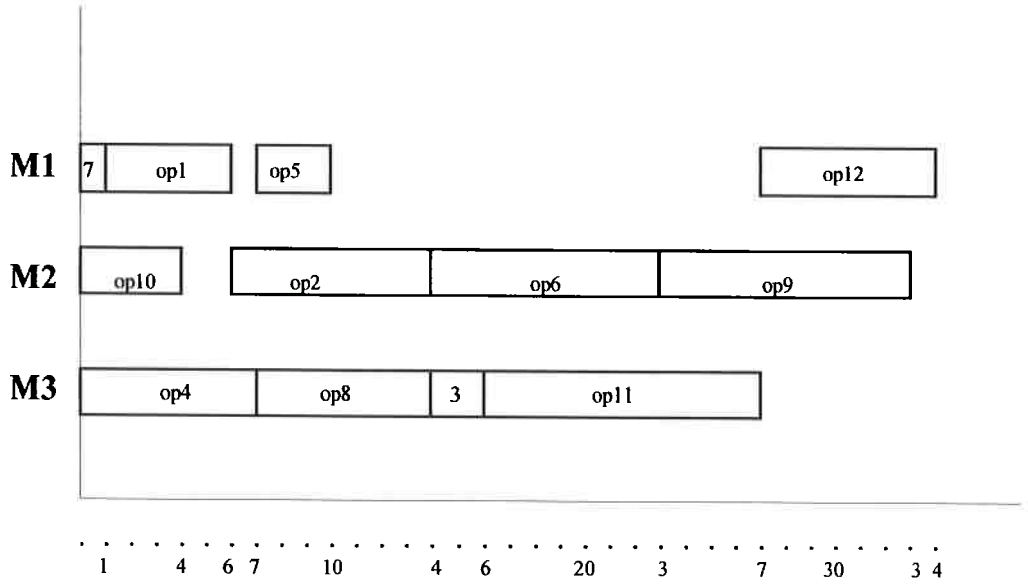


Figura I.12 - A Solução do Planejador baseado RdP para problema da Tabela I.2

I.4.2 - Problema exemplo da Tabela I.3

Agora, vamos considerar o problema definido na Tabela I.3. O Menu do sistema tem a função para *Ler_Arquivo_Processos*. Nesta função, a definição da RdP que descreve o domínio do problema é lida na forma de termos Visual PROLOG gravados em arquivo. O arquivo ksk_plus.rdp contem os termos correspondentes à Tabela I.3.

Em seguida, deve-se definir as ordens de fabricação e os recursos disponíveis. Por exemplo: definimos demandas para: p1, p2, p3 e p4 com todos os recursos: m1, m2 e m3 disponíveis. Ativando a função Solucionar do Menu, temos a solução que está listada em seguida. Esta solução tem uma duração ("makespan") de 18 horas.
A listagem a seguir, ilustra o plano por recursos:

Listagem do Plano por Recursos: Data: 18/11/1996 Hora: 14:26:36:92

i) Recurso: m1

tempo: 0	inicia: op8	recurso alocado: m1	duração: 3	término: 3
tempo: 3	termina: op8	recurso liberado: m1	início: 0	duração: 3
	inicia: op4	recurso alocado: m1	duração: 6	término: 9
tempo: 9	termina: op4	recurso liberado: m1	início: 3	duração: 6
tempo: 10	inicia: op2	recurso alocado: m1	duração: 5	término: 15
tempo: 15	termina: op2	recurso liberado: m1	início: 10	duração: 5

	inicia: op10	recurso alocado: m1	duração: 2	término: 17
tempo: 17	termina: op10	recurso liberado: m1	início: 15	duração: 2

ii) Recurso: m2

tempo: 0	inicia: op5	recurso alocado: m2	duração: 3	término: 3
tempo: 3	termina: op5	recurso liberado: m2	início: 0	duração: 3
	inicia: op12	recurso alocado: m2	duração: 3	término: 6
tempo: 6	termina: op12	recurso liberado: m2	início: 3	duração: 3
	inicia: op1	recurso alocado: m2	duração: 4	término: 10
tempo: 10	termina: op1	recurso liberado: m2	início: 6	duração: 4

iii) Recurso: m3

tempo: 0	inicia: op7	recurso alocado: m3	duração: 3	término: 3
tempo: 3	termina: op7	recurso liberado: m3	início: 0	duração: 3
	inicia: op11	recurso alocado: m3	duração: 4	término: 7
tempo: 7	termina: op11	recurso liberado: m3	início: 3	duração: 4
	inicia: op9	recurso alocado: m3	duração: 6	término: 13
tempo: 13	termina: op9	recurso liberado: m3	início: 7	duração: 6
	inicia: op6	recurso alocado: m3	duração: 3	término: 16
tempo: 16	termina: op6	recurso liberado: m3	início: 13	duração: 3
	inicia: op3	recurso alocado: m3	duração: 2	término: 18
tempo: 18	termina: op3	recurso liberado: m3	início: 16	duração: 2

A Figura I.13, seguinte, ilustra o plano listado acima para o problema definido na Tabela I.3 e elaborado pelo Planejador baseado em Rede de Petri.

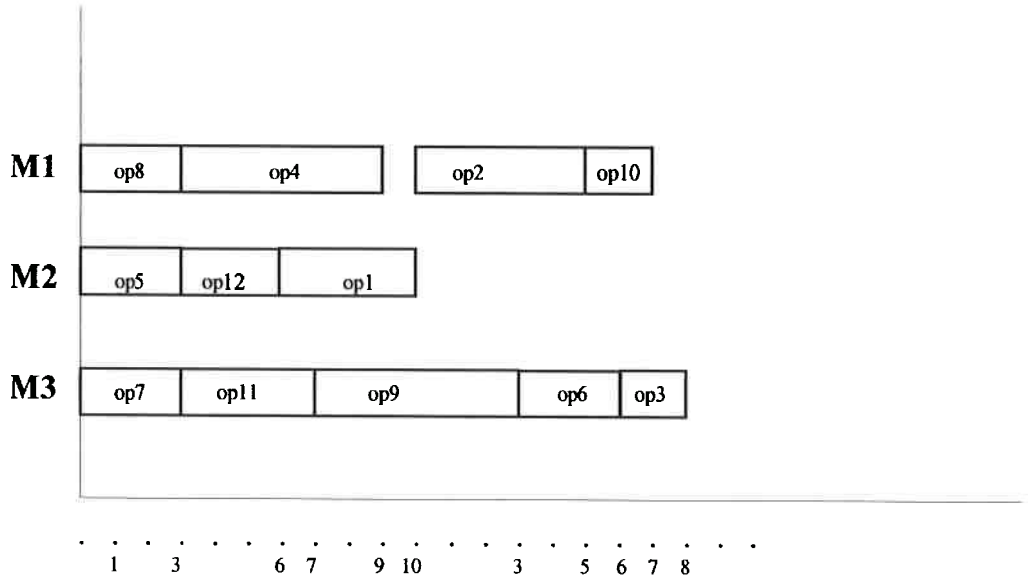


Figura I.13 - A Solução do Planejador baseado RdP para problema da Tabela I.3

Este sistema é muito flexível porque pode tratar os roteiros de produção alternativos para as operações foram definidos na Tabela I.3, como mostra a Figura I.11.a. O sistema pode encontrar uma solução sempre que pelo menos duas máquinas estiverem operando, i.e., quando uma determinada máquina estiver inoperante. Por exemplo, suponha que existam ordens de fabricação para: p1, p2, p3 e p4 e existem os recursos: m1 e m3. Em seguida, apresentamos a listagem gerada pelo sistema para este problema. Devido a utilização de roteiros alternativos de produção para contornar o fato do recurso m2 estar inoperante, a duração do plano gerado aumentou de 18 para 28 horas.

Listagem do Plano por Recursos: Data: 18/11/1996 Hora: 16:23:42:60

i) Recurso: m1

tempo: 0	inicia: op8	recurso alocado: m1	duração: 3	término: 3
tempo: 3	termina: op8	recurso liberado: m1	início: 0	duração: 3
	inicia: op12	recurso alocado: m1	duração: 5	término: 8
tempo: 8	termina: op12	recurso liberado: m1	início: 3	duração: 5
	inicia: op4	recurso alocado: m1	duração: 6	término: 14
tempo: 14	termina: op4	recurso liberado: m1	início: 8	duração: 6
	inicia: op6	recurso alocado: m1	duração: 5	término: 19
tempo: 19	termina: op6	recurso liberado: m1	início: 14	duração: 5
	inicia: op2	recurso alocado: m1	duração: 5	término: 24
tempo: 24	termina: op2	recurso liberado: m1	início: 19	duração: 5
	inicia: op3	recurso alocado: m1	duração: 4	término: 28
tempo: 28	termina: op3	recurso liberado: m1	início: 24	duração: 4

ii) Recurso: m3

tempo: 0	inicia: op7	recurso alocado: m3	duração: 3	término: 3
tempo: 3	termina: op7	recurso liberado: m3	início: 0	duração: 3
	inicia: op5	recurso alocado: m3	duração: 4	término: 7
tempo: 7	termina: op5	recurso liberado: m3	início: 3	duração: 4
	inicia: op11	recurso alocado: m3	duração: 4	término: 11
tempo: 11	termina: op11	recurso liberado: m3	início: 7	duração: 4
	inicia: op1	recurso alocado: m3	duração: 6	término: 17
tempo: 17	termina: op1	recurso liberado: m3	início: 11	duração: 6
	inicia: op9	recurso alocado: m3	duração: 6	término: 23
tempo: 23	termina: op9	recurso liberado: m3	início: 17	duração: 6
	inicia: op10	recurso alocado: m3	duração: 2	término: 25
tempo: 25	termina: op10	recurso liberado: m3	início: 23	duração: 2

A Figura I.14, seguinte, ilustra o plano listado acima para o problema modificado com m2 inoperante.

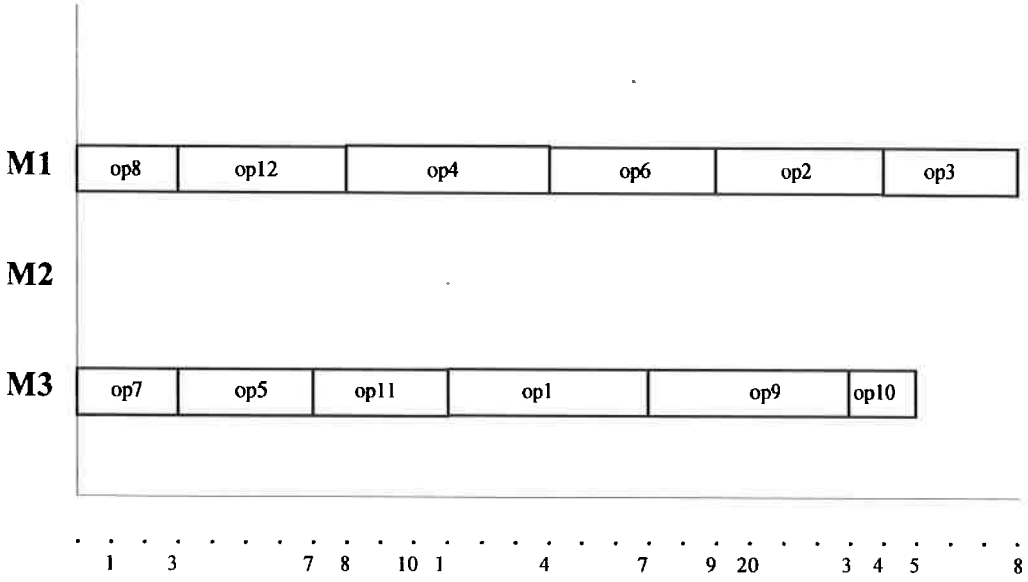


Figura I.14 A Solução do Planejador para o problema modificado

I.5 - Conclusão

O desempenho computacional obtido nesta implementação foi bastante razoável. A solução descrita no item I.4.1, na página 15, gerada para o problema exemplo definido no item I.1 (Tabela I.2), foi encontrada pelo Planejador baseado em Rede de Petri dentro de um tempo de processamento de 93 centésimos de segundo -- tempo apenas para a elaboração da solução -- usando um microcomputador tipo IBM PC 486 DX2 de 33 Mhz. Repetindo o bom desempenho, a solução descrita no item I.4.2, na página 17, gerada para o outro problema exemplo definido no item I.2 (Tabela I.3), foi encontrada pelo Planejador baseado em Rede de Petri dentro de um tempo de processamento 1 segundo e 26 centésimos de segundo, usando o mesmo microcomputador.

Com esses resultados práticos obtidos, podemos concluir que a implementação da metodologia proposta mostrou ser uma solução eficiente e sem necessidade de fazer "backtracking" na falha para problemas de planejamento em sistemas dinâmicos, i.e., com qualidade assegurada desde o princípio.

* * * * * * * * *

REFERÊNCIAS BIBLIOGRÁFICAS

[Araribóia 1989], Grupo ARARIBÓIA. Inteligência artificial: um curso prático. Livros Técnicos e Científicos Ltda. 1989. Rio de Janeiro.

[Clocksin et Mellish 1987] CLOCKSIN W.F., MELLISH C.S., Programming in Prolog. 1987. Springer-Verlag. Germany.

[Fox 1984], FOX, M.; SMITH, S.F. ISIS - a knowledge-based system for factory scheduling. Expert Systems, July 1984, Vol.1, n.1. In: ALLEN, J.; HENDLER, J.; TATE, A., coord. **Readings in planning**. San Mateo, California. Morgan Kaufmann, 1990. Cap. 5, p.336-360.

[Kusiak 1990], KUSIAK, Andrew. Intelligent manufacturing systems. Prentice-Hall, Englewood Cliffs. 1990. New Jersey.

[Miyagi 1990], MIYAGI, P.E. Modelagem e controle de sistemas produtivos - aplicação da teoria de redes de petri. Monografias do Departamento de Engenharia Mecânica nº. 55, 1990. Escola Politécnica, Universidade de São Paulo.

[PDC 1996] Prolog Development Center A / S, Visual PROLOG Version 4.0 - Language Tutorial, 1996. Denmark.

[Sterling et Shapiro 1986] STERLING, L. SHAPIRO E., The Art of Prolog - Advanced Programming Techniques. 1986. The MIT Preess. USA.

[Silva et Shimad 1995] SILVA, J. Reinaldo, SHIMADA, L. M. Um Sistema de Planejamento Semi-Reativo de Sistemas Produtivos Dinâmicos usando Redes de Petri. 2° Simpósio Brasileiro de Automação Inteligente. 13 a 15.set.1995. CEFET-PR. Curitiba.

[Valette et Silva 1989] VALETTE, R., SILVA. M. Petri Nets and Flexible Manufacturing. Lecture Notes in Computer Sciences. vol. 424. Springer-Verlag. 1989.

APÊNDICE II

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

II UM "PLANNING-SCHEMATA" USANDO RdP

Vamos descrever como a proposta descrita no capítulo 4, pode ser aplicada para se desenvolver "planning-schemata" usando-se a Rede de Petri. Em nossa proposta o modelo de ação é representado através de uma rede de Petri que constitui o modelo conceitual (*mc*) do problema. O *mc* consiste de redes de Petri para representar o domínio da aplicação do problema de "planning" ("application model") e o problema de "planning" em si ("planner self model"). No cap. 4, mostramos como a análise das estrutura da rede de Petri provê o meta-conhecimento necessário para dirigir o processo de "planning", através do exemplo do Mundo dos Blocos.

Apresentamos o Sistema Implementado para o Mundo de Blocos, no cap. 4, item 4.8. A aplicação escrita em Visual., Já vimos que, o Menu de Barras para esta aplicação apresenta dois modos para se acionar o Planejador: modo *Rede de Petri* (o plano não é interpretado para a linguagem do domínio) e modo *Mundo de Blocos* (o plano é interpretado para a linguagem do domínio). É indiferente para o sistema, o modo como o Planejador é acionado -- o mecanismo de inferência (MI) sempre irá resolver o problema da mesma forma, i.e., levando em conta a rede de Petri.

A implementação do sistema para o Mundo de Blocos, descrito no cap. 4, foi feita na linguagem Visual PROLOG para Windows 3.11 [PDC 1996], em dois módulos separados:

- módulo planejador baseado em Rede de Petri, e
- módulo de interface com o usuário.

O módulo planejador, basicamente, implementa o Planejador baseado em Rede de Petri. O plano é elaborado incrementalmente, usando-se um ambiente de redes de Petri que opera sobre a rede que representa o conhecimento do domínio. Conseqüentemente, todas as ordens de serviço são escalonadas em paralelo.

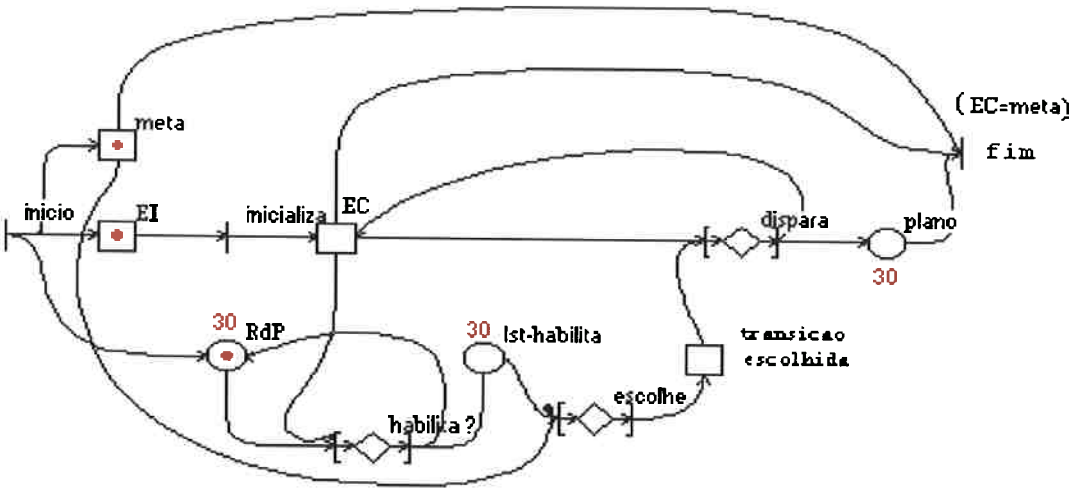


Figura II.1 - A RdP estendida Ghenesys que representa o "planning self model"

A cada passo, o Planejador elabora uma lista de transições habilitadas. É utilizado um critério_corrente para a escolha da transição. Então, faz-se o disparo da transição, atualizando todos os estados da RdP, passando-se, então, ao passo seguinte. Para o problema do item 4.8, este critério é o do caminho mínimo. A Fig. II.1, acima, ilustra o "planning self model" para o Mundo de Blocos, usando a noção de caminho mínimo.

A seguir, vamos descrever como o mesmo módulo planejador baseado em Rede de Petri pode ser reutilizado para resolver vários problemas de "planning" localizados em outro domínio.

Tabela II.1 - Distâncias entre cidades (milhas)

Origem/ Destino	Carlisle (car)	Darlington (dlr)	Newcastle (nwc)	Penrith (pen)	townA (twA)	townB (twB)	townC (twC)	Workington (wrk)
Carlisle	0		58	23				33
Darlington		0	40	52	25			
Newcastle	58	40	0					
Penrith	23	52		0				39
townA		25			0	15		
townB					15	0	10	
townC						10	0	5
Workington	33			39			5	0

II.1 O Problema do Caminho Mínimo

O Problema do Caminho Mínimo é um problema clássico de travessia em grafos [Cloksin et Mellish 1985]. Consiste em se encontrar a menor distância entre dois nós de um grafo não direcionado, fortemente conexo. Seja o problema de caminho mínimo, descrito na Tab. II.1, acima.

A Figura I.2, seguinte, ilustra a rede de Petri C / E que representa o "application model" para o problema descrito na tabela II.1. A distância entre os lugares está indicado nas transições (milhas).

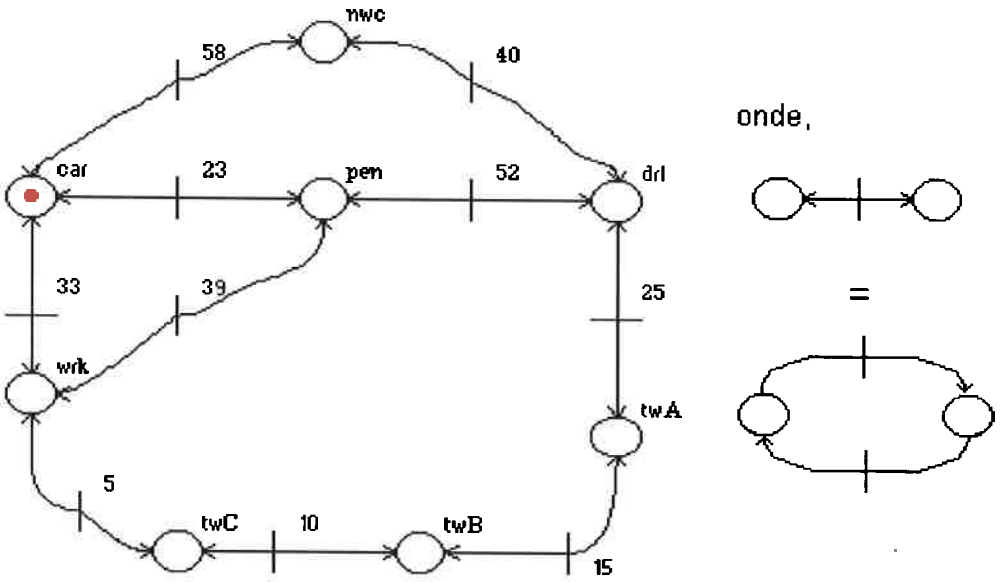


Figura II.2 - "Application model" para problema de caminho mínimo

Podemos reutilizar módulo planejador baseado em Rede de Petri elaborado para o problema do Mundo de Blocos, ilustrado na Fig. II.1, que implementa a noção de caminho mínimo para resolver este problema. Desta forma, apenas o módulo de módulo de interface com o usuário é que deve ser construído para este problema. Para isso, a noção de caminho mínimo, apresentada no item 4.8.1., deve ser modificada para:

caminho mínimo: seqüência que corresponde à menor soma de distâncias possível para se movimentar a marca de um determinado lugar de origem para outro lugar de destino.

Figura I.3 - Menu de Barras para o Problema do Caminho Mínimo

A interface do programa mantém a mesma funcionalidade do programa para o Mundo de Blocos: o Menu de Barras tem o modo *Rede-de-Petri* e o modo interpretado para o domínio do problema *Caminho-Minimo*, como mostra a Figura I.3, acima. O problema pode ser definido tanto no modo *Rede-de-Petri*, assim como no modo interpretado. Inicializando o problema, definimos o estado inicial como sendo car e o estado meta como sendo twA, usando o modo *Rede-de-Petri*.

O Problema do Caminho Minimo

condicao inicial: car
condicao meta: twA

Se usarmos, agora, o modo *Caminho-Minimo* do Menu de Barras, podemos ter a interpretação destes estados da rede de Petri para a linguagem que descreve o domínio:

condicao inicial: carlisle
condicao meta: townA

Planejador baseado em Rede de Petri opera da seguinte maneira. Inicialmente, o sistema está no estado inicial: car. Daí, tem-se uma lista de transições habilitadas. O sistema avalia a distância total a partir do estado corrente até a meta desejada, para cada transição habilitada, como mostramos a seguir:

candidato: car-nwc dist. :123
candidato: car-pen dist. :92
candidato: car-wrk dist. :63

Como o critério é o caminho mínimo, dispara-se a transição "car-wrk" que corresponde ao menor caminho. Daí, tem-se o estado corrente "wrk". Repetindo-se o procedimento acima, tem-se:

candidato: wrk-pen dist. :108
candidato: wrk-twC dist. :35

A transição escolhida é "wrk-twC", que tem o menor caminho estimado. Daí, o estado corrente é "twC". Repetindo, novamente, o procedimento, tem-se:

```
candidato: twC-twB   dist. :30
```

A transição escolhida é "twC-twB", que tem o menor caminho estimado. Daí, o estado corrente é "twB". Repetindo, novamente, o procedimento, tem-se:

```
candidato: twB-twA   dist. :15
```

A transição escolhida é "twB-twA", que já leva até o estado meta desejado. Deste modo, temos o plano completo:

```
O Problema do Caminho Minimo

 condicao inicial: car
 condicao meta:  twA

atividade:  car-wrk     resultado:  wrk
atividade:  wrk-twC     resultado:  twC
atividade:  twC-twB     resultado:  twB
atividade:  twB-twA     resultado:  twA
```

Listando o plano acima, interpretado para o domínio do problema fica:

```
O Problema do Caminho Minimo

 condicao inicial: carlisle
 condicao meta: townA

acao:  de carlisle para workington     resultado: workington
acao:  de workington para townC        resultado: townC
acao:  de townC para townB             resultado: townB
acao:  de townB para townA             resultado: townA
```

II.2 Um Esquema para Resolver Problemas Combinatórios

A abordagem da Inteligência Artificial (IA) tem uma atração especial na resolução de problemas combinatórios, porque permite obter rapidamente uma solução dentre as potencialmente inúmeras possíveis soluções. A dificuldade com este tipo de problema está na necessidade de realizar raciocínio "não-monotônico". O Planejador tem de saber diferenciar ações aparentemente indiferentes quanto ao resultado alcançado, mas que podem conduzir a "dead-ends" e requerer, por este motivo, a realização de

"backtracking". Este tipo de estruturação é comparada com o tipo de representação a partir dos "primeiros princípios", i.e., da representação de ações primitivas, sem diferenciar o contexto no qual a ação está sendo aplicado. No capítulo 4, ilustramos como o uso do modelo conceitual baseado em rede de Petri pode ser valioso. No exemplo do Mundo de Blocos, ao invés de fazer a busca usando métodos ingênuos como o "meios-fins", que pode conduzir a "dead-ends", nossa proposta utiliza mecanismos auxiliares para estruturar o problema. Através da análise da estrutura da rede de Petri, conseguimos elaborar uma estratégia de busca da solução capaz de evitar a Anomalia de Sussman.

Vamos mostrar, através de exemplos que a abordagem proposta pode ser boa para desenvolver sistemas que caracterizam-se como problemas combinatórios. Os exemplos são três: problema das jarras, o problema do fazendeiro e o problema dos missionários e canibais.

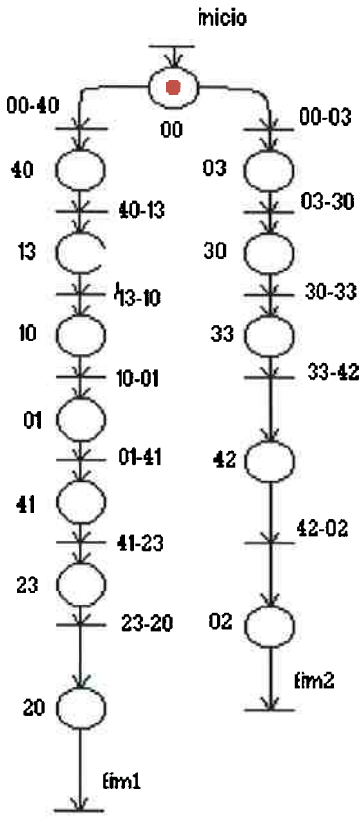


Figura II.4 - "Application model" para o problema das jarras

II.2.1 - O Problema das Jarras

O Problema das Jarras é outro exemplo de problema combinatório que pode ser resolvido pela abordagem proposta. Tem-se uma fonte ("source") e um sorvedouro ("sink") de água e duas jarras não graduadas, mas com capacidade conhecida: uma jarra de 4 litros e outra jarra de 3 litros. O objetivo é o de se obter exatamente 2 litros, indiferentemente, em qualquer das duas jarras.

Observe que uma abordagem baseada nos primeiros princípios, por exemplo, associado com a estratégia "meios-fins" pouco ajuda na estruturação do problema. Neste problema, existem seis ações primitivas para se tentar atingir o objetivo. Estas são: (1) encher a jarra de 4 l, (2) encher a jarra de 3 l, (3) esvaziar a jarra de 4 l, (4) esvaziar a jarra de 3 l, (5) transferir a jarra de 4 l para a jarra de 3 l, e (6) transferir a jarra de 3 l para a jarra de 4 l. Na Figura II.4, na página anterior, mostramos como a modelagem do "application model" em rede de Petri C/E, ajuda a estruturar o problema, explicitando corretamente o efeito da aplicação de uma ação.

A Tabela II.2, seguinte, mostra a correspondência entre as seis ações primitivas e as transições da rede de Petri C/E mostrada na Figura II.4. Observe que para uma mesma ação primitiva, existem mais de uma transição correspondentes. Por exemplo, para a ação primitiva (1) "encher a jarra de 4 l na fonte", pode corresponder tanto à transição "00-40" quanto à transição "01-41", ou seja, a ambas tem a mesma interpretação.

Tabela II.2 - Transições da RdP correspondentes às ações primitivas

Número	Descrição da ação primitiva	Transições da RdP
1	Encher a jarra de 4 l	"00-40", "01-41"
2	Encher a jarra de 3 l	"00-03", "30-33"
3	Esvaziar jarra de 4 l	"42-02"
4	Esvaziar jarra de 3 l	"13-10", "23-20"
5	Transferir jarra de 4 l para 3 l	"40-13", "10-01", "41-23"
6	Transferir jarra de 3 l para 4 l	"03-30", "33-42"

O Planejador baseado em Rede de Petri é o mesmo utilizado para o Problema do Mundo de Blocos, apenas desconsiderando-se o critério de caminho mínimo para a esco-

lha da transição. Aqui, uma estratégia puramente "forward" é capaz de encontrar uma solução para o problema, porque a RdP é uma árvore, existindo conflito (daí, a necessidade de se fazer uma escolha) apenas entre as transições "00-40" e "00-03", a partir da condição inicial "00" . A RdP da Figura II.4 foi elaborada levando em conta apenas estados factíveis do problema. Por este motivo, o sistema sequer considera a possibilidade de ocorrer estados não factíveis na solução do problema.

Figura II.5 - Menu de barras para o Problema da Jarras

A Figura II.5, acima, ilustra o Menu de barras para o Problema das Jarras. O Planejador pode ser acionado, indiferentemente, em dois modos: *Dominio-RdP* e *Domínio-Jarras*. Fazendo-se a inicialização do problema no modo *Dominio-RdP*, informando como estado inicial o estado "00", tem-se:

```
O problema do jarras

condicao inicial:  00
condicao final:    02
```

Este estado inicial, no modo *Domínio-Jarras*, corresponde ao estado:

```
O problema do jarras

condicao inicial:  jarra 4l vazia, jarra 3 l vazia
condicao final:    jarra 4 l vazia, jarra 3 l com 2 l
```

Resolvendo-se o problema para o *Domínio-RdP*, temos o plano seguinte:

```
O problema do jarras

condicao inicial:  00
condicao final:    02

passos:
00, 00-03, 03-30, 30-33, 33-42, 42-02
```

Este plano, interpretado para o *Domínio-Jarras*, fica:

O problema do jarras

condicao inicial: jarra 4l vazia, jarra 3 l vazia
condicao final: jarra 4 l vazia, jarra 3 l com 2 l

ação: Encher a jarra de 3 l
ação: Transferir jarra de 3 l para 4 l
ação: Encher a jarra de 3 l
ação: Transferir jarra de 3 l para 4 l
ação: Esvaziar a jarra de 4 l

Observe que o Planejador baseado em Rede de Petri não usa em nenhum momento a interpretação do domínio do problema para obter a solução do plano. Ao contrário, ele opera exclusivamente sobre a RdP. O *Domínio-Jarras* serve apenas como uma interface para o Usuário. Desta forma, o Usuário pode navegar entre os dois modos de operação, indiferentemente, a todo instante. O Planejador sempre irá traduzir os comandos para o *Domínio-RdP*.

O Planejador é capaz de encontrar soluções alternativas, se estas existirem. Para isto, basta acionar a função "*Outra-Solução*" do Menu de barras, em qualquer modo de operação. Fazendo isto para a solução corrente, tem-se:

Outra Solução

O condicao inicial: 00
O condicao final: 20

passos:
00-40, 40-13, 13-10, 10-01, 01-41, 41-23, 23-20

A interpretação desta solução alternativa para o *Domínio-Jarras*, fica:

O problema do jarras

condicao inicial: jarra 4l vazia, jarra 3 l vazia
condicao final: jarra 4 l com 2 l, jarra 3 l vazia

ação: Encher a jarra de 4 l
ação: Transferir jarra de 4 l para 3 l
ação: Esvaziar a jarra de 3 l
ação: Transferir jarra de 4 l para 3 l
ação: Encher a jarra de 4 l
ação: Transferir jarra de 4 l para 3 l
ação: Esvaziar a jarra de 3 l

II.2.2 O Problema do Fazendeiro

Um fazendeiro deve levar um lobo, um carneiro e um repolho para a outra margem do rio. Em cada travessia somente uma das três cargas, além do barqueiro, pode ser carregada na balsa.

Represente a organização de transporte sob a condição de nunca poderem ser deixados a sós em qualquer uma das margens: (a) o lobo e o carneiro (b) o carneiro e o repolho [Miyagi 1993a]. Como o problema é relativamente pequeno, podemos fazer uma enumeração completa dos estados, como mostra a Tabela II.3, seguinte.

Tabela II.3 - Enumeração completa do Problema do Fazendeiro

Estado	fazendeiro	lobo	carneiro	repolho	factível
0	norte	norte	norte	norte	sim
1	norte	norte	norte	sul	sim
2	norte	norte	sul	norte	sim
3	norte	norte	sul	sul	não
4	norte	sul	norte	norte	sim
5	norte	sul	norte	sul	sim
6	norte	sul	sul	norte	não
7	norte	sul	sul	sul	não
8	sul	norte	norte	norte	não
9	sul	norte	norte	sul	não
A	sul	norte	sul	norte	sim
B	sul	norte	sul	sul	sim
C	sul	sul	norte	norte	não
D	sul	sul	norte	sul	sim
E	sul	sul	sul	norte	sim
F	sul	sul	sul	sul	sim

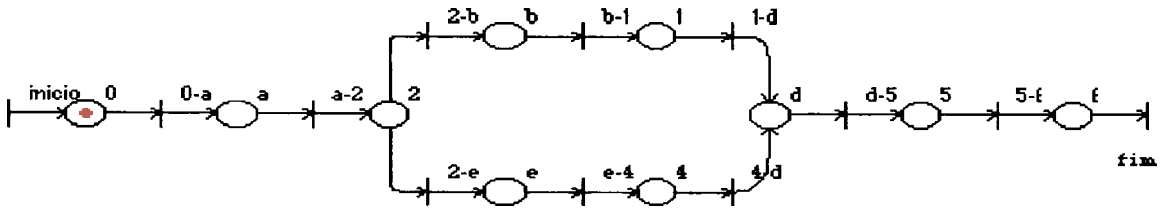


Figura II.6 "Application model" para o Problema do Fazendeiro

Mapeando em rede de Petri C / E, apenas os estados factíveis da Tab. II.3, acima, temos a RdP que representa o domínio da aplicação (ou seja, o "application model"), considerando-se apenas os estado factíveis. O "application model" para o Problema do Fazendeiro está ilustrado na Figura II.6, acima.

A Tabela II.4, seguinte, ilustra a descrição das transições da rede de Petri C/E mostrada na Fig. II.6, acima, interpretado para o domínio do problema.

Observe na Tabela II.4, que as ações primitivas podem ter mais de uma transição que a representa. Isto porque o efeito de sua aplicação não é indiferente quanto ao estado do mundo no qual se aplicam as ações. Por exemplo, as transições "0-a" e "5-f" correspondem, ambas, à ação interpretada no domínio como "O fazendeiro leva o carneiro". O papel da rede de Petri é, justamente, de explicitar estas diferenças.

Tabela II.4 - Interpretação das transições para o domínio

Transição	Descrição no domínio
"0-0"	O estado inicial
"0-a"	O fazendeiro leva o carneiro
"a-2"	O fazendeiro volta sozinho
"2-e"	O fazendeiro leva o lobo
"e-4"	O fazendeiro volta como carneiro
"4-d"	O fazendeiro leva o repolho
"2-b"	O fazendeiro leva o repolho
"b-1"	O fazendeiro volta com o carneiro
"1-d"	O fazendeiro leva o lobo
"d-5"	O fazendeiro volta sozinho
"5-f"	O fazendeiro leva o carneiro
"f-f"	O estado final desejado

O Planejador baseado em Rede de Petri, praticamente, pode ser o mesmo utilizado para o Problema das Jarras, descrito no item II.2.1. Uma estratégia puramente "forward" é capaz de encontrar uma solução para o problema, porque a RdP da Figura II.6 tem apenas um conflito real (ou existência de marcas que exigem uma necessidade de escolha) entre as transições "2-b" e "2-e", a partir do estado "2" (existe outro conflito estrutural entre "1-d" e "4-d", mas, como esta rede é uma máquina de estado, o conflito nun-

ca se torna um conflito real). A RdP foi construída levando em conta apenas os estados factíveis. Deste modo, partindo-se a partir de um estado inicial factível, não há nenhuma razão para que ocorra uma perda da factibilidade durante o processo de busca da solução.

A exemplo da aplicação para o Problema do Caminho Mínimo, o módulo de interface com o Usuário, apresenta Menu de Barras com dois modos de operação: modo *Rede-de-Petri* e Modo *Problema-do-Fazendeiro*. A Figura II.7, a seguir, ilustra o menu de barras para esta aplicação.

Figura II.7 Menu de barras para o Problema do Fazendeiro

Vamos usar o modo *Rede-de-Petri* para inicializar o programa, informando o estado inicial = estado "0" (o estado meta é sempre fixo = estado "f"). Resolvendo o problema, tem-se o plano:

```
O problema do fazendeiro

 condicao inicial: 0
 condicao final:   f


passos:
0-0, 0-a, a-2, 2-e, e-4,
4-d, d-5, 5-f, f-f
```

Para obter uma interpretação da solução para o domínio, usar o modo Problema-do-Fazendeiro e acione a função *Mostrar-Solução*. A listagem a seguir corresponde à interpretação do plano acima.

```
O problema do fazendeiro

 condicao inicial: ["norte","norte","norte","norte"]   -   estado "0"
 condicao final:   ["sul","sul","sul","sul"]           -   estado "f"

norte   norte   norte   norte   ação:  O estado inicial informado
norte   norte   norte   norte   ação:  O fazendeiro leva o carneiro
sul     norte   sul     norte   ação:  O fazendeiro volta sozinho
norte   norte   sul     norte   ação:  O fazendeiro leva o lobo
sul     sul     sul     norte   ação:  O fazendeiro volta com carneiro
norte   sul     norte   norte   ação:  O fazendeiro leva o repolho
sul     sul     norte   sul     ação:  O fazendeiro volta sozinho
norte   sul     norte   sul     ação:  O fazendeiro leva o carneiro
sul     sul     sul     sul     ação:  O estado final desejado
```

O sistema verifica se existe uma outra solução para este problema. Neste caso, ele falha para o plano corrente e faz um "backtracking" na rotina de planejamento. Fazendo isto, obtemos a solução a seguir.

```
Outra Solução

 condicao inicial: 0
 condicao final:   f

passos:
0-0, 0-a, a-2, 2-b, b-1,
1-d, d-5, 5-f, f-f
```

A solução alternativa, quando interpretada para o domínio, fica:

```
norte   norte   norte   norte   ação:  O estado inicial informado
norte   norte   norte   norte   ação:  O fazendeiro leva o carneiro
sul     norte   sul     norte   ação:  O fazendeiro volta sozinho
norte   norte   sul     norte   ação:  O fazendeiro leva o repolho
sul     norte   sul     sul     ação:  O fazendeiro volta com carneiro
norte   norte   norte   sul     ação:  O fazendeiro leva o lobo
sul     sul     norte   sul     ação:  O fazendeiro volta sozinho
norte   sul     norte   sul     ação:  O fazendeiro leva o carneiro
sul     sul     sul     sul     ação:  O estado final desejado
```

II.2.3 O Problema dos Missionários e Canibais

O Problema dos Missionários e Canibais é uma variante do Problema do Fazendeiro, descrito no item II.2.2. Três missionários devem atravessar levando três canibais num barco até a outra margem do rio. A condição de segurança dos missionários é a de que nunca o número de canibais deve ser superior ao número de missionários em nenhum instante.

Novamente, o módulo Planejador baseado em Rede de Petri pode ser o mesmo usado para o Problema das Jarras, descrito no item II.2.1. O módulo de interface com o usuário deve prover a Rede de Petri C/E que representa o domínio da aplicação (ou seja, o "application model").

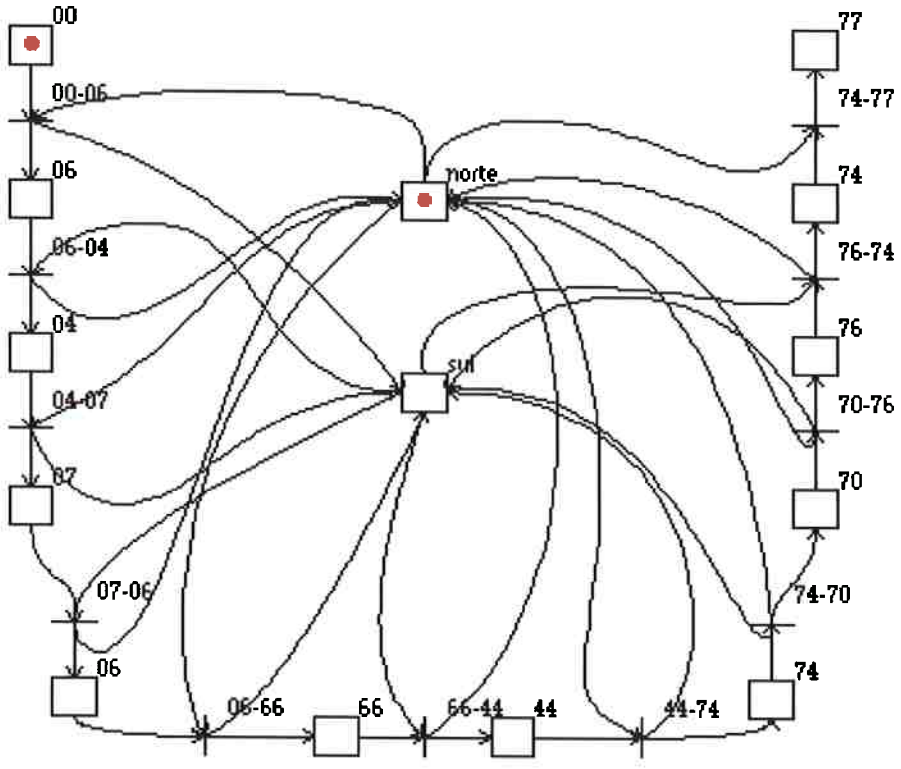


Figura II. 8 - "Application model" para o problema dos Missionários e Canibais

A Figura II.8, acima, ilustra a rede de Petri C / E que representa o domínio da aplicação para o problema dos Missionários e Canibais. Neste caso, não podemos fazer o tipo de simplificação feita no Problema do Fazendeiro em que a localização do barco

era redundante com a localização do fazendeiro. Portanto, a localização do barco deve ser explicitada. Aqui, utilizamos uma modelagem semelhante ao usado para representar o uso de recursos produtivos nos Sistemas Flexíveis de Fabricação - SFF, descrita no Apêndice I. A Tabela II.5 a seguir, ilustra a interpretação dos lugares representados na rede de Petri C / E, da Figura II.8.

Tabela II.5 - Interpretação dos lugares da RdP C / E da Figura II.8

Lugar	binário	mission1	mission2	mission3	canibal1	canibal2	canibal3
"00"	0000 0000	norte	norte	norte	norte	norte	norte
"04"	0000 0100	norte	norte	norte	sul	norte	norte
"06"	0000 0110	norte	norte	norte	sul	sul	norte
"07"	0000 0111	norte	norte	norte	sul	sul	sul
"44"	0100 0100	sul	norte	norte	sul	norte	norte
"66"	0110 0110	sul	sul	norte	sul	sul	norte
"70"	0111 0000	sul	sul	sul	norte	norte	norte
"74"	0111 0100	sul	sul	sul	sul	norte	norte
"76"	0111 0110	sul	sul	sul	sul	sul	norte
"77"	0111 0111	sul	sul	sul	sul	sul	sul

As Tabelas II.6.a e II.6.b, a seguir, apresentam a interpretação das transições da rede de Petri C / E da Figura II.8, respectivamente, para o barco na margem norte e para o barco na margem sul.

Tabela II.6.a Barco na margem norte

Transição	Descrição
"00-06"	Atravessam 2 canibais
"04-07"	Atravessam 2 canibais
"06-66"	Atravessam 2 missionários
"44-74"	Atravessam 2 missionários
"70-76"	Atravessam 2 canibais
"74-77"	Atravessam 2 canibais

Tabela II.6.b Barco na margem sul

Transição	Descrição
"06-04"	Volta 1 canibal
"07-06"	Volta 1 canibal
"07-04"	Voltam 2 canibais
"66-44"	Voltam 1 missn. e 1 canibal
"74-70"	Volta 1 canibal
"76-74"	Volta 1 canibal

Novamente, observe que as transições nas Tabelas II.6.a-b podem corresponder a uma mesma ação primitiva. Por exemplo, existem quatro transições da RdP que correspondem, todas elas, à mesma ação primitiva de "atravessam dois canibais".

Figura II.9 - Menu de barras para o problema dos Missionários e Canibais

A Figura II.9, acima, ilustra o menu de barras da aplicação. Aqui, o menu de barras também apresenta dois modos de operação: modo *Domínio-RdP* e modo *Domínio-Missionários*. Vamos usar o modo *Domínio-Missionarios* para *Inicializar* o problema. Suponha que informamos o estado inicial "07" (o estado final é fixo sempre como sendo o estado "77").

O problema do missionarios

condicao inicial: ["norte","norte","norte","sul","sul","sul"] barco: sul
condicao final: ["sul","sul","sul","sul","sul","sul"] barco: sul

Obtendo-se a solução deste problema interpretado para o domínio do problema de Missionários e Canibais, o plano fica:

O problema do missionarios

m1	m2	m3	c1	c2	c3	barco	ação
norte	norte	norte	sul	sul	sul	sul	A condicao inicial informada
norte	norte	norte	sul	sul	sul	sul	Volta um canibal
norte	norte	norte	sul	sul	norte	norte	Atravessam dois missionarios
sul	sul	norte	sul	sul	norte	sul	Voltam 1 mission. e 1 canibal
sul	norte	norte	sul	norte	norte	norte	Atravessam dois missionarios
sul	sul	sul	sul	norte	norte	sul	Volta um canibal
sul	sul	sul	norte	norte	norte	norte	Atravessam dois canibais
sul	sul	sul	sul	sul	norte	sul	Volta um canibal
sul	sul	sul	sul	norte	norte	norte	Atravessam dois canibais
sul	sul	sul	sul	sul	sul	sul	A condicao final desejada

Neste exemplo, vamos fazer o processo inverso, listando agora, a solução em termos de rede de Petri. O sistema não impor nenhuma ordem em particular, os modos de operação podem ser intercambiados indiferentemente , a todo instante, porque o Planejador sempre opera em termos de RdP. Fazendo isso, temos:

O problema do missionarios

condicao inicial: 07 barco: 1
condicao final: 77 barco: 1

passos:
00-00, 07-06, 06-66, 66-44, 44-74,
74-70, 70-76, 76-74, 74-77, 77-77

II.3 - O Problema do Mundos de Blocos Generalizado

Agora, vamos mostrar como podemos estender a solução para o Problema do Mundo de Blocos [Fikes et Nilsson 1971], [Sussman 1974], mostrada no cap. 4, para qualquer número de blocos, usando a metodologia proposta.

Partindo do fato de que, para um número de blocos (NB) fixo, o número de estados possíveis é limitado, concluímos é possível se calcular analiticamente diversas variáveis do problema em função de NB como, por exemplo, o número de níveis N *k* -- onde k é definido através do número mínimo de passos para se atingi-lo a partir o nó essencial (NE) -- o número de pilhas não vazias de blocos ("stacks") em cada nível *s*, etc.

Notação para representar os estados:

A notação para representar os estados é feita através de listas de listas. As listas internas representam stacks onde o bloco mais profundo corresponde à posição do stack. Se o bloco que corresponde à posição do stack não estiver sobre a mesa, o stack é vazio. Por exemplo: para NB=4, o estado bla sobre mesa, blb sobre bla, blc sobre mesa, bld sobre blc é representado por [[blb,bla,[],[bld,blc],[]].

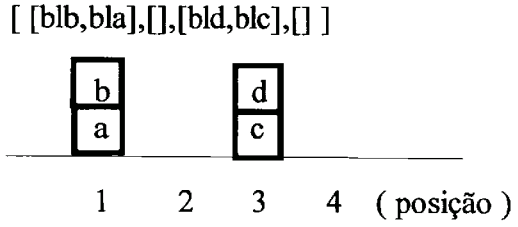


Figura II.10 - Exemplo de Representação para NB = 4

Definições:

NE - nó essencial é o estado onde é verdadeiro que todos os blocos estão sobre a mesa e é verdadeiro que todos os blocos estão livres.

N_k = (ou Nível k) - conj. de estados com número mínimo de k passos para se atingir o nó essencial (NE).

k = NB - s *(k = 0,1,...,NB-1)*

s = quantidade de stacks não vazios nos estados do Nível *(s= 1,2,..,NB)*

$k = \min_i \{ r_i \}$ tal que $i \in N k$

r_i = número de passos para se atingir um estado i, $(i = 1,...,n)$ a partir do nó essencial (NE), onde n = número de estados atingíveis do problema.

Observe que, para um determinado NB fixo, o número de níveis da rede de Petri do tipo C / E que representa o problema é igual a k = NB - s, onde s é o número de pilhas de blocos ("stacks") localizadas no Nível N k que não estão vazias. k também pode ser visto como o número mínimo de passos para atingir o nível N k a partir do NE.

O número de blocos em cada stack é dado por $I j$. O vetor $\mathbf{I}_k$ representa uma lista de todos os stacks $I j$ no Nível N_k. Dentro do conjunto de estados atingíveis n para NB blocos, o vetor $\mathbf{I}_k$ é dado pela expressão:

$$\mathbf{I}_k = \{ I_k \} \mid \sum_{j=1}^{s} I_j = NB \quad I \in \{ 1,2,3,...,NB \}$$

Ilustramos na Tabela II.7, à seguir, mostra como pode variar o número de blocos em cada stack I_j do vetor $\mathbf{I}_k$, situado no nível N_k para o problema do Mundo de Blocos com NB = 6,5,4, e 3 blocos.

Tabela II. 7 - Configurações Possíveis dos Blocos em função do NB

I_k	NB = 6	NB = 5	NB = 4	NB = 3
Nivel N_0	[1,1,1,1,1,1]	[1,1,1,1,1]	[1,1,1,1]	[1,1,1]
Nivel N_1	[2,1,1,1,1,0]	[2,1,1,1,0]	[2,1,1,0]	[2,1,0]
Nivel N_2	[3,1,1,1,0,0], [2,2,1,1,0,0]	[3,1,1,0,0], [2,2,1,0,0]	[3,1,0,0], [2,2,0,0]	[3,0,0]
Nivel N_3	[4,1,1,0,0,0], [3,2,1,0,0,0], [2,2,2,0,0,0]	[4,1,0,0,0], [3,2,0,0,0]	[4,0,0,0]	[0,0,0]
Nível N_4	[5,1,0,0,0,0], [4,2,0,0,0,0], [3,3,0,0,0,0]	[5,0,0,0,0]	[0,0,0,0]	[0,0,0]
Nível N_5	[6,0,0,0,0,0]	[0,0,0,0,0]	[0,0,0,0]	[0,0,0]

A Tabela II.7 ilustra apenas a soma número de blocos que aparece em cada pilha, em cada configuração. Nada podemos afirmar acerca da distribuição dos blocos que estão empilhados uns sobre os outros. Observe que o número máximo de níveis *max* k = NB - 1. Por exemplo, para NB = 6, *max* k = 5, i.e., o maior nível é $N_k = N_5$. Ainda, considerando NB = 6, para o nível $N_k = N_0$, a quantidade de stacks não vazios $s = 6$ (o que confirma a fórmula k = NB - s, ou seja 0 = 6 - 6). No nível $N_k = N_1$, a quantidade de stacks não vazios $s = 5$, confirmando a fórmula analítica que define que k = NB - s, ou seja 1 = 6 - 5.

Os estados do problema da Tabela II.7, podem ser mapeados para grafos bipartidos da Redes de Petri C / E. A análise da estrutura da RdP resultante pode prover "insigths" extremamente úteis a partir das quais podemos derivar algoritmos de busca eficientes. Por exemplo, podemos observar que:

1) o estado onde todos os blocos estão sobre a mesa corresponde a um nó essencial (NE) da RdP. O NE da RdP é um estado que não pertence à nenhuma subrede em particular mas, pode ser incluído, indistintamente, à qualquer das subredes formando um subconjunto estendido de transições que são interdependentes.
2) o primeiro bloco movido a partir do nó essencial determina uma sucessão de estados dependentes desta ação e que correspondem à uma mesma componente conectada da RdP.

3) se os dois estados EI e Meta pertencem ambas à mesma componente conectada, então, pertencem a um conjunto de transições que são interdependentes entre si, mas, independentes com respeito ao restantes das transições da RdP. Portanto, o plano pode ser elaborado observando apenas a subrede que contem estes dois estados, e nada mais.
4) o plano envolvendo dois estados em duas componentes conectadas diferentes pode ser resolvido estabelecendo-se como submeta o restabelecimento do nó essencial.
5) uma estratégia de busca judiciosa baseada em RdP pode evitar o problema da Anomalia de Sussman ao discriminar corretamente as conseqüências da aplicação de ações primitivas.

Para entender melhor a observação 4, acima, devemos lembrar a noção de processos independentes de [Georgeff 1987]. No cap. 4, mostramos que, partindo-se do nó essencial (NE) em que todos os blocos estão sobre a mesa, independente do número de blocos (NB) envolvido, o primeiro bloco que é movimentado sobre outro bloco qualquer define uma relação de dependência com todos os estados que lhe sucedem. Deste modo, o primeiro bloco movimentado define uma subrede, ou seja, uma mesma componente conectada formada por um conjunto de transições interdependentes, mas, que independem de qualquer outra transição da rede de Petri. Esta noção de subrede torna-se importante porque se EI e M pertencem ambas à mesma subrede, nenhuma outra subrede precisa ser levada em consideração para a elaboração do plano.

A Observação 1 acima indica que o NE da RdP é um lugar que pode pertencer a qualquer das subredes, indistintamente. Portanto, se EI e M não pertencem à mesma subrede, o NE é um candidato por excelência para compor a meta intermediária. Deste modo, podemos dividir o problema em dois subproblemas que compõem processos que são independentes, o sistema elabora um plano parcial como solução para cada um deles. O plano final será obtido a partir da concatenação destes planos parciais.

Feitas estas observações, agora, podemos generalizar o algoritmo para o Mundo de Blocos, para NB = 3 blocos, descrito no cap. 4, item 4.3.1, para um número inteiro positivo qualquer de número de blocos.

II.3.1 - O Algoritmo para n blocos

São dados: NB = número de blocos,
EI = estado inicial e
M = conjunção de metas.

Algoritmo para n blocos:

0) Agrupar blocos segundo os níveis N*k* em que se situam no estado Meta M desejado e ordenar estes níveis em ordem crescente, i.e. a partir do Nível *0* até o Nível *k-1*.
1) SE M já é verdadeiro ENTÃO fim.
2) EI e/ou M é nó essencial (NE).
Nó essencial = (bl1 sobre mesa, bl2 sobre mesa, ... , blNB sobre mesa).
2.1) SE EI é nó essencial ENTÃO resolva M usando uma estratégia do tipo "forward chaining"
2.2) SE M é nó essencial ENTÃO resolva M usando uma estratégia do tipo "forward chaining"
3) SE EI e M estão na mesma componente conectada ENTÃO ∃ caminho entre EI e M sem passar pelo nó essencial. Resolva M usando uma estratégia do tipo "forward chaining".

Obs: Como foi anteriormente, os estados que compõem uma mesma componente conectada estão definidos pelo predicado seguinte:

```
MESMA_COMPONENTE_CONECTADA(EI,M) :-
                        PERTENCE(S,[a,b,c,d,...]),
                        PERTENCE_COMPO_SX(EI,S),
                        PERTENCE_COMPO_SX(M,S).
PERTENCE_COMPO_SX((Estado,S) :-
                    PERTENCE(X sobre mesa, Estado),
                    PERTENCE(S sobre X, Estado).
```

4) CASO CONTRÁRIO, EI e M estão localizados em subredes distintas. Gere uma submeta MI = nó essencial obtendo um subplano PI. À seguir, resolva M a partir

do nó essencial usando uma estratégia "forward chaining" obtendo um subplano P2. O plano total P = PI + P2.

O algoritmo acima não é capaz de obter a solução ótima, i.e., não pode encontrar o caminho mínimo. Mas a sua vantagem está em que, assim, o Planejador pode raciocinar monotonicamente, ou seja, pode elaborar o plano sem trocar de estratégia, qualquer que seja o número de blocos NB envolvido.

Para obter a solução para uma mesma subrede, o Planejador deve enfrentar três situações:

i) atingir o estado meta a partir do nó essencial (NE). A solução pode ser encontrada de uma forma trivial, apenas posicionando os blocos, que estão todos livres, obedecendo a ordem parcial dos blocos existente no estado meta desejado.

ii) atingir o nó essencial (NE) a partir de qualquer estado inicial. A solução pode ser encontrada de uma forma trivial, apenas posicionando sobre a mesa os blocos, que estão livres no topo de cada pilha existente, até atingir o estado desejado NE.

ii) atingir o estado meta M, a partir de um EI, sem passar pelo nó essencial (NE). A solução nessa caso é mais complexa. O NE não precisa ser restabelecido porque, eventualmente, alguma pilha de blocos pode não precisar ser totalmente desfeita. Uma possível estratégia que combina uma execução parcial dos passos (i) e (ii) acima pode ser pensada. Todos os blocos que não estão na posição desejada devem ser tornados livres, movendo-os para a mesa. Para isso, basta identificar pilhas de blocos de EI que já tem elementos na posição desejada no estado meta M, começando do bloco mais profundo.

Implementamos o algoritmo acima usando a linguagem Arity / PROLOG for DOS, Version 6.0 [Arity 1988]. A seguir, ilustramos alguns resultados obtidos no uso deste programa. A vantagem do uso do algoritmo acima está em não se ter a necessidade de elaborar explicitamente a RdP que representa o "application model". Ao invés, as propriedades desta RdP são obtidas analiticamente através de fórmulas.

II.3.1.1 - O Sistema Implementado

Inicialmente, seja o caso onde, NB = 5 blocos. Vamos partir do estado inicial EI = NE = [[bl1], [bl2], [bl3], [bl4], [bl5]] e vamos definir como meta M = [[bl2,bl1], [], [bl5, bl4, bl3], [], []].

```
?- strips.
  inicio;
  bl2 sobre bl1;
  bl4 sobre bl3;
  bl5 sobre bl4.
  ->
  yes
```

Agora, Vamos manter a solução acima como sendo o estado inicial EI e definir uma meta situada em outra subrede, M = [[], [bl3,bl2], [], [bl5, bl1, bl4], []]. EI pertence às subredes S_2 e S_4, enquanto M pertence às subredes S_1 e S_3. Observe que o plano está claramente dividido em duas partes, a primeira parte das ações restabelece o nó essencial e a segunda parte constrói as pilhas para alcançar o estado meta M desejado.

```
?- strips.
  inicio;
  bl2 sobre mesa;
  bl5 sobre mesa;
  bl4 sobre mesa;
  bl1 sobre bl4;
  bl3 sobre bl2;
  bl5 sobre bl1.
  ->
  yes
```

Finalmente, vamos fazer um teste para o caso em que EI e M estão na mesma subrede. Seja o EI igual ao estado final da solução acima. E seja M = [[], [bl1, bl2], [bl4, bl3], [], [bl5]]. Nesse caso, EI e M, ambas, pertencem à subrede comum S_1.

```
?- strips.
  inicio;
  bl3 sobre mesa;
  bl5 sobre mesa;
  bl1 sobre bl2;
  bl4 sobre bl3.
  ->
  yes
```

Observe que o programa encontrou a solução correta para NB = 5 blocos. Agora, vamos fazer o teste para NB = 10 blocos. Vamos partir do estado inicial EI = NE e M = [[], [bl3,bl2], [], [bl5, bl1, bl4], [], [], [bl8, bl7], [], [bl10, bl6, bl9], []]. A solução fica:

```
strips.
inicio;
bl1 sobre bl4;
bl3 sobre bl2;
bl6 sobre bl9;
bl8 sobre bl7;
bl5 sobre bl1;
bl10 sobre bl6.
->
yes
```

Agora, Vamos manter a solução acima como o estado inicial EI e definir uma meta situada em outra subrede, M = [[bl2,bl1], [], [bl5, bl4, bl3], [], [], [bl7, bl6], [], [bl10, bl9, bl8], [], []].

```
?- strips.
 inicio;
 bl3 sobre mesa;
 bl5 sobre mesa;
 bl8 sobre mesa;
 bl10 sobre mesa;
 bl1 sobre mesa;
 bl6 sobre mesa;
 bl2 sobre bl1;
 bl4 sobre bl3;
 bl7 sobre bl6;
 bl9 sobre bl8;
 bl5 sobre bl4;
 bl10 sobre bl9.
 ->
 yes
```

Finalmente, vamos fazer um teste para o caso em que EI e M estão na mesma subrede. Seja o EI igual ao estado final da solução acima. E seja M = [[], [bl1, bl2], [bl4, bl3], [], [bl5], [], [bl6, bl7], [bl9, bl8], [], [bl10]].

```
?- strips.
 inicio;
 bl2 sobre mesa;
 bl5 sobre mesa;
 bl7 sobre mesa;
 bl10 sobre mesa;
 bl1 sobre bl2;
 bl6 sobre bl7.
 ->
 yes
```

Observe que o programa encontrou a solução correta para NB = 10 blocos. Em seguida, vamos ver como podemos melhorar ainda mais o desempenho do algoritmo descrito no item II.3.1., passando a considerar outros resultados que podem ser obtidos a partir da análise da estrutura da rede de Petri.

II.3.2 Identificando as Submetas "Short-Cuts"

Para o caso particular onde NB = 3 blocos, a Figura 37, no cap. 4, mostra três subredes bem definidas, S_A, S_B e S_C, tendo como único estado comum o nó essencial (NE). Observe na Tabela II.6, na coluna para NB = 3, que, é claro que, não existe nenhum vetor $\mathbf{I}_k$ que apresenta duas colunas com número de blocos maior ou igual a dois -- que poderiam definir lugares da RdP que pertencem a mais de uma subrede. Portanto, a única meta intermediária que podemos utilizar para planejar movimentos envolvendo duas componentes conectadas distintas é o nó essencial (NE).

Considere, agora, o caso onde NB = 4 blocos. A Figura II.11, seguinte, ilustra a rede de Petri C / E que representa o "application model" para este problema, que foi elaborado usando o ambiente "Petri Net für Windows" [Frank et Schmidt 1993]. A Figura II.11, mostra agora quatro subredes: S_A, S_B, S_C e S_D. Note que entre as subredes S_A e S_B existem agora dois lugares comuns na rede de Petri: N.E. e o lugar "5". Este lugar "5", corresponde ao conjunto de estados que tem a configuração [[blX, bla], [blY, blb], [], []]. Note que, abusando da notação, usamos blX e blY não como variáveis a instanciar, mas, sim, como "place holder" significando que não importa qual seja a instanciação feita. Uma notação mais rigorosa, aqui, seria importante para se distinguir

melhor este conjunto de estados. Por exemplo, o lugar "5" corresponde aos seguintes estados atômicos: [[], [], [bla, blc], [blb, bld]] e [[], [], [blb, blc], [bla, bld]].

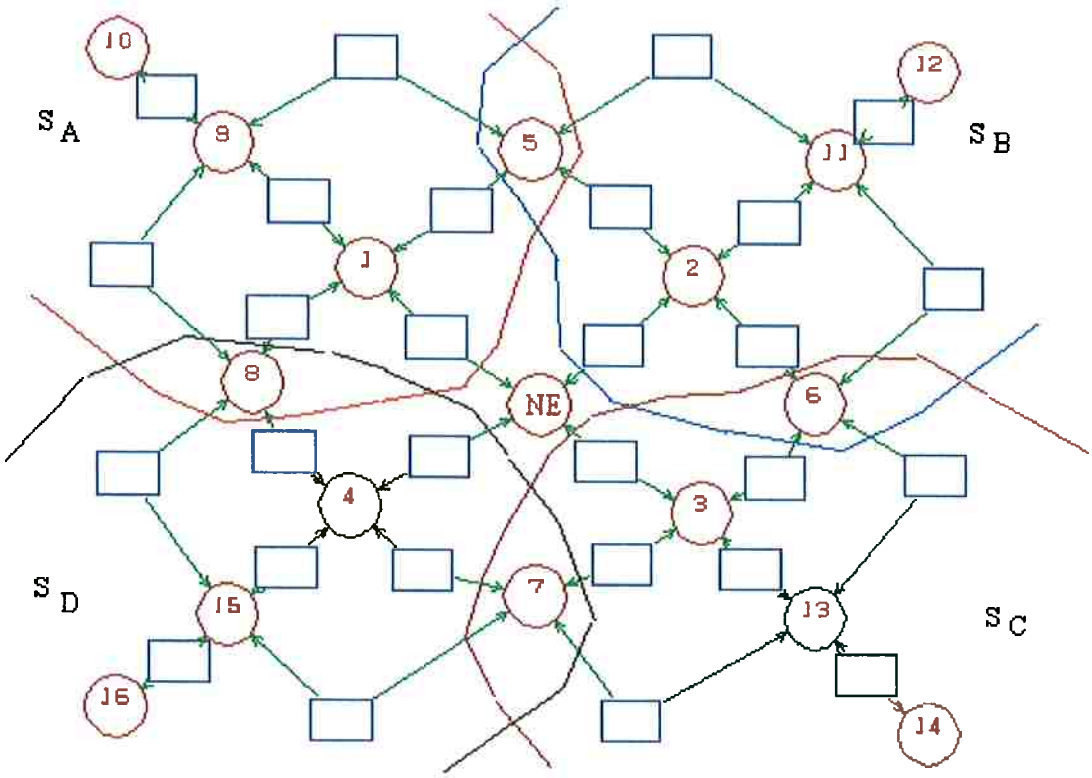


Figura II.11 - "Application model" para Mundo de Blocos, NB = 4,

Os lugares da rede de Petri na Fig. II.11, agora, não representam um só estado, mas sim um conjunto de estados que apresentam uma determinada configuração dos blocos e está situado em determinado nível N *k*. Esta configuração é, exatamente, a configuração que está na Tabela II.6, para a coluna NB = 4.

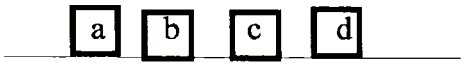
A Tabela II.8, seguinte, ilustra a configuração dos lugares na rede de Petri da Figura II.11, para NB = 4 blocos. Observe que, neste caso, além do NE, os lugares "5", "6", "7", e "8" pertencem a mais de uma subrede simultaneamente. Estes lugares coincidem, justamente, com o vetor $\mathbf{I}_k = [2,2,0,0]$ para o nível N_2 na Tabela II.6. Portanto, para NB = 4, estes lugares NE, "5", "6", "7", e "8" podem constituir a meta intermediária referida na Obs.4, na página II.20.

Tabela II.8 - Configuração dos lugares na rede de Petri da Figura II.11

Lugar	Configuração	Subrede
NE	[[bla], [blb], [blc], [bld]]	A, B, C, D
1	[[blX,bla], [blY], [blZ], []]	A
2	[[blX,blb], [blY], [blZ], []]	B
3	[[blX,blc], [blY], [blZ], []]	C
4	[[blX,bld], [blY], [blZ], []]	D
5	[[blX,bla], [blY,blb], [], []]	A, B
6	[[blX,blb], [blY,blc], [], []]	B, C
7	[[blX,blc], [blY,bld], [], []]	C, D
8	[[blX,bld], [blY,bla], [], []]	A, D
9	[[blX,bla,blY], [blZ], [], []]	A
10	[[blX,bla,blY,blZ], [], [], []]	A
11	[[blX,blb,blY], [blZ], [], []]	B
12	[[blX,blb,blY,blZ], [], [], []]	B
13	[[blX,blc,blY], [blZ], [], []]	C
14	[[blX,blc,blY,blZ], [], [], []]	C
15	[[blX,bld,blY], [blZ], [], []]	D
16	[[blX,bld,blY,blZ], [], [], []]	D

O NE para NB = 4 blocos, está ilustrado na Figura II.12, seguinte.

[[bla], [blb], [blc],[bld]]



1 2 3 4 (posição)

Figura II.12 Estado 0 (NE) para NB = 4 blocos

A configuração dos lugares representados pelo vetor $\mathbf{I}_k = [2,2,0,0]$ para o nível N_2 está ilustrado na Figura II.13.a-d,seguinte, que são os lugares "5", "6", "7" e "8".

[[bla,blX],[],[blb,blY],[]]

Fig. II.13.a Lugar "5"

[[blb,blX],[],[blc,blY],[]]

Fig. II.13.b Lugar "6"

[[blc,blX],[],[bld,blY],[]]

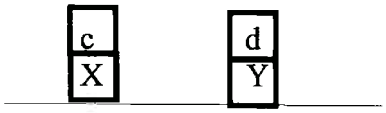


Fig. II.13.c Lugar "7"

[[ble,blX],[],[blf,blY],[]]

Fig. II.13.b Lugar "8"

Como mostramos acima, este mesmo procedimento poderá ser generalizado, por indução, para ser aplicado para o resolver o problema com qualquer número de blocos NB. Para isto, basta construir a Tabela II.6 e repetir o mesmo tipo de raciocínio.

Vamos ilustrar isto, usando o caso onde NB = 5 blocos. Podemos ver na Tabela II.6 que maior nível *max k* = 5 - 1 = 4, ou seja, N_4 é o maior nível para NB = 5. Para se estabelecer metas intermediárias como prevê a Obs.4, da página II.24, devemos procurar as configurações que apresentam mais de duas pilhas de blocos com numero de blocos maior ou igual a dois. Observando a tabela II.6, podemos identificar a configuração [2,2,1,0,0] localizada no nível N_2 e a configuração [3,2,0,0,0] localizada no nível N_3.

Portanto, a Observação 4, na página II.20, pode ser alterada para o seguinte texto:

4) o plano envolvendo dois estados em duas componentes conectadas diferentes pode ser resolvido estabelecendo-se como submeta:
   4.1) o restabelecimento do nó essencial, ou
   4.2) um lugar (ou estado) comum entre duas submetas conectadas diferentes. Um determinado lugar da RdP pertence a mais de uma componente conectada quando possui mais de duas pilhas de blocos ("stacks") cuja representação em lista contém mais de dois blocos.

Conseqüentemente, o algoritmo para o Mundo de Blocos Generalizado para qualquer número de blocos, descrito no item II.3.1 pode ser alterado para:

II.3.3 - O Algoritmo usando Submetas "Short-Cuts")

São dados: NB = número de blocos,
EI = estado inicial e
M = conjunção de metas.

Algoritmo para n blocos:

0) Agrupar blocos segundo os níveis N*k* em que se situam no estado Meta M desejado e ordenar estes níveis em ordem crescente, i.e. a partir do Nível *0* até o Nível *k-1*.
1) SE M já é verdadeiro ENTÃO fim.
2) EI e/ou M é nó essencial (NE).
Nó essencial = (bl1 sobre mesa, bl2 sobre mesa, ... , blNB sobre mesa).
2.1) SE EI é nó essencial ENTÃO resolva M usando uma estratégia do tipo "forward chaining"
2.2) SE M é nó essencial ENTÃO resolva M usando uma estratégia do tipo "forward chaining"
3) SE EI e M estão na mesma componente conectada ENTÃO ∃ caminho entre EI e M sem passar pelo nó essencial. Resolva M usando uma estratégia do tipo "forward chaining".
4) CASO CONTRÁRIO, EI e M estão localizados em subredes distintas.
4.1) gere uma submeta MI = lugar comum entre as duas subredes, obtendo um subplano PI. Vá para passo 4.3. Se não existir este lugar comum, então
4.2) gere uma submeta MI = nó essencial obtendo um subplano PI.
4.3) À seguir, resolva M a partir do nó essencial usando uma estratégia "forward chaining" obtendo um subplano P2. O plano total P = PI + P2.

Partindo do fato de que, para um número de blocos (NB) fixo, o número de estados possíveis é limitado. Concluímos, finalmente, que é possível se calcular analiticamente diversas variáveis do problema em função de NB. A vantagem do uso deste algoritmo está em não se ter a necessidade de elaborar explicitamente a RdP que representa o "application model". Ao invés, as propriedades desta RdP são obtidas analiticamente através de fórmulas.

Desta maneira, temos um algoritmo que é capaz de resolver usando uma estratégia que é monotônica (ou seja, que não se modifica quaisquer que sejam os estados inicial, final ou corrente) para resolver um problema reconhecidamente não monotônico como é o problema do Mundo de Blocos. Além disso, porque analisa (dinamicamente no momento de resolver o problema) a estrutura da RdP C/E que representa o "application model" através de fórmulas analíticas, ao invés de elaborar explicitamente a RdP, o algoritmo pode resolver o problema do Mundo de Blocos para qualquer número de blocos.

* * * * * * * * *

REFERÊNCIAS BIBLIOGRÁFICAS

[Arity 1988] Arity Corp. The Arity / PROLOG - Language Reference Manual. 1988. Concord, MA, USA.

[Clocksin et Mellish 1987] CLOCKSIN W.F., MELLISH C.S., Programming in Prolog. 1987. Springer-Verlag. Germany.

[Fikes et Nilsson 1971], 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.

[Frank et Schmidt 1993] FRANK, M, SCHMIDT, V. Petri Netze für Windows, Version 2.0 1993. Germany.

[Georgeff 1987], GEORGEFF, M.P. Planning. Ann. Rev. Comput. Sci. 1987. Id. Ibid. Cap.1, p.5-25.

[Miyagi 1993a], MIYAGI, P.E. Introdução ao projeto de sistemas por redes de petri. Livro-texto, 1993. Curso de Engenharia Mecatrônica. Escola Politécnica. Universidade de São Paulo.

[PDC 1996] Prolog Development Center A / S, Visual PROLOG Version 4.0 - Language Tutorial, 1996. Denmark.

[Sussman 1974], 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.

APÊNDICE III

[i]Um Sistema de Planejamento Semi-Reativo de Sistemas Produtivos Dinâmicos usando Redes de Petri

Shimada, Lúcio Mitio[1] **Silva, José Reinaldo**[2]
[1] E-mail: a707@dirinf-sp.petrobras.com.br
[2] E-mail: reinaldo@usp.br

[1,2] Laboratório de Automação e Sistemas, Departamento de Engenharia Mecânica, Escola Politécnica da Universidade de São Paulo

Resumo

As Redes de Petri (RdP) são largamente utilizadas na modelagem do controle de sistemas dinâmicos por eventos discretos. Para se obter um melhor desempenho computacional na solu ção de problemas de planejamento, é necessário se adotar mecanismos auxiliares para estruturar o problema. Com este objetivo, usamos as RdP. Mostramos que a análise estrutural de uma RdP pode prover uma estrutura para problemas de síntese em sistemas de eventos discretos. Por exemplo, esta análise é capaz de sugerir estratégias para uma rápida busca de solução minimizando e, em alguns casos, eliminando o "backtracking". Para ilustrar isto, usamos o exemplo clássico do Mundo de Blocos.

Nosso trabalho enfatiza as redes aplicativas PFS/MFG, derivadas da RdP e está dirigido a aplicações em sistemas flexíveis de fabricação (SFF). O modelo conceitual do problema está expresso em níveis de abstração distintos. A idéia é a de que se queremos modificar com uma certa freqüência o plano, uma maneira de faze-lo selecionando adequadamente o nível de abstração, de acordo com a complexidade e a exiguidade do tempo de resposta. Trabalhamos dentro dos limites onde as mudanças exigidas não modificam drasticamente o plano original.

Apresentamos uma implementação simples do "scheduling" de um Sistema Flexível de Fabricação (SFF) usando a linguagem PROLOG e um formalismo de Redes de Petri Condição/Evento para se fazer uma modelagem estrutural.

Palavras-chave: Redes de Petri, inteligência artificial, planejamento, sistemas dinâmicos, MFG/PFS.

1. O Planejamento de tarefas na IA

A maioria dos sistemas de planejamento utiliza uma representação do tipo STRIPS [Fikes et Nilsson 1971]. Nesta, a descrição de uma ação apresenta: uma pré-condição, uma lista de adição e uma lista de eliminação. Os sistemas aplicam ações para modificar o estado do mundo se suas pré-condições estiverem verdadeiras. O efeito da aplicação de uma ação está expresso nas listas de adição e nas listas de eliminação. Exemplo, a Fig. 1, a seguir, mostra o problema do Mundo de Blocos.

GOAL: ACHIEVE (AND (ON A B) (ON B C))

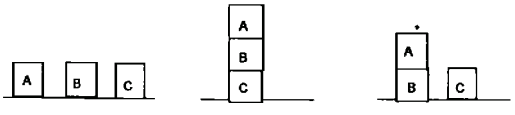


Figura 1 - O Mundo de Blocos

Os sistemas de planejamento utilizam bastante conhecimento implícito e incompleto, o que torna este tipo de problema muito complexo.

Para se obter um melhor desempenho computacional na solução de problemas de planejamento, é necessário se adotar mecanismos auxiliares para estruturar o problema. Com este objetivo, usamos a teoria das Redes de Petri (RdP) para a construção de um modelo conceitual do problema, visando a identificação de restrições que possam auxiliar na solução do problema de planejamento. Nosso trabalho enfatiza as redes aplicativas PFS/MFG aplicadas ao "design" e planejamento de sistemas flexíveis de fabricação (SFF).

Representações baseadas em grafos sempre estiveram presentes no tratamento do problema de planejamento. Entre os textos clássicos, [Sacerdoti

1974] utiliza árvores de busca e [Tate 1977], utiliza uma representação de grafo de estados para planos. Em trabalhos mais recentes, [Drummond 1985], [Hendler 1992], já utilizam grafos bipartidos derivados das Redes de Petri.

Em uma breve cronologia da aplicação de grafos em planejamento temos:

- No sistema NOAH, planos são representados como redes procedimentais (scripts) [Sacerdoti 1974] onde as propriedades estruturais do problema não são representadas.

- [Tate 1977] foi o primeiro a introduzir redes de PERT na resolução do problema de planejamento (sistema NONLIN).

- [Drummond 1985] utilizou uma representação para o plano, derivada da rede de Petri (RdP). A rede final se assemelha a uma RdP do tipo Lugar/Transição, Entretanto, as regras de disparo de transições não levam em conta as pós-condições. Além disso, a rede de Petri usada não usa nenhum tipo de hierarquia.

- [Hendler 1992] especifica os aspectos procedimentais dos operadores de uma forma semelhante às redes de Petri do tipo C/E. Entretanto, do mesmo modo que Drummond, as regras de disparo de transições não levam em conta as pós-condições. Além disso, a comunicação entre níveis é feita através de sistemas de tradução localizadas abaixo de cada nível, quebrando o formalismo das RdP. Em nossa proposta, o PFS/MFG faz a integração total do sistema produtivo e do sistema de informações numa única arquitetura hierárquica, como mostraremos no item 5, a seguir. O modelo conceitual é constituído por diversos níveis hierárquicos de abstração, usando ferramentas de uma mesma família (PFS/MFG).

2. Uma estrutura simples baseada em grafos

A RdP permite a representação formal dos estados e da evolução de um sistema de eventos discretos. Permite explicitar relações entre estes estados, por ex. dependência de ordem parcial entre estados, a existência de eventos concorrentes ou excludentes, compartilhamento de recursos, etc.

A análise estrutural de uma RdP pode prover informação importante para uma abordagem não-procedural (baseada em IA) em sistemas de eventos discretos. O conhecimento sobre a estrutura do problema pode subsidiar formas de busca mais eficazes, e a minimização do "backtracking". Como uma primeira abordagem para este problema, mostraremos a seguir o tratamento de um problema clássico em planejamento, o planejamento de ações no mundo dos blocos. A escolha se justifica pela simplicidade do problema em si em contraposição com o fato de apresentar todas as características que dificultam a automação do planejamento.

2.2 O Exemplo do Mundo de Blocos

Seja uma instância particular do Mundo de Blocos, composta de três blocos aleatoriamente distribuídos sobre uma "mesa".

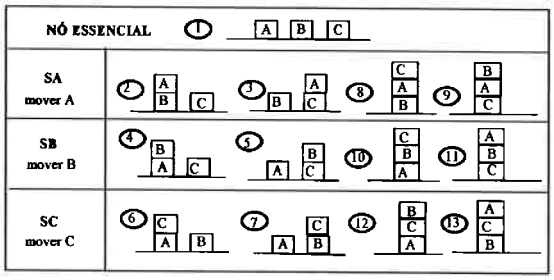


Figura 2 Estados do Mundo de Blocos

A Figura 2, acima, mostra todos os atingíveis a partir do estado "inicial" designado aqui de "nó essencial". A Figura 3, a seguir, mostra a representação em RdP do tipo C/E que constitui o modelo conceitual deste problema. Analisando a estrutura da RdP, podemos observar que:

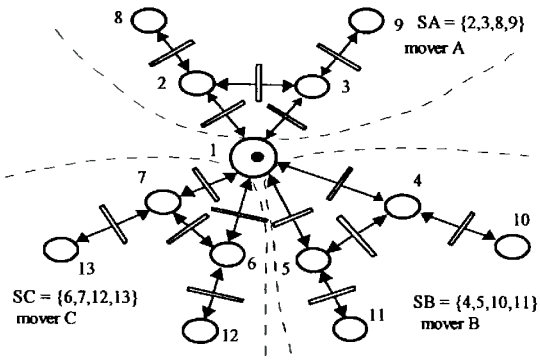


Figura 3 RdP C/E para o Mundo de Blocos

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

Fazendo uso deste "conhecimento" poderemos, por exemplo, evitar o problema da anomalia de Sussman.

anomalia de Sussman [Sussman 1974]: 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 pro-

grama entra num processo de "looping" sem progresso em direção ao estado meta desejado.
Um plano para se atingir um estado alvo a partir do estado inicial poderia ser dado pelo seguinte algoritmo: Sejam dados EI = estado inicial e M = conjunção de metas.

1) SE M já é verdadeiro ENTÃO fim.
   Ex. M = (a sobre b $\wedge$ b sobre c)
2) SE EI e/ou M é nó essencial (NE) ENTÃO resolva M usando mecanismo de backward chaining.
3) SE EI e M estão na mesma componente conectada ENTÃO $\exists$ caminho entre EI e M sem passar pelo nó essencial. Resolva M usando mecanismo de backward chaining.
4) Caso Contrário, gere uma submeta MI = nó essencial obtendo um subplano P1. À seguir, resolva M a partir do nó essencial usando mecanismo de backward chaining obtendo um subplano P2. O plano total P = P1 + P2.

Evidentemente, para problemas reais em FMS, uma representação em RdP do tipo C/E apresenta o inconveniente da falta de expressividade. Devemos esperar que problemas reais resultem em grafos muito grandes onde a complexidade de uma análise computacional cresce exponencialmente. Mesmo em sistemas flexíveis para pequena e média empresa, e portanto menores, uma análise estrutural é proibitiva. Portanto procuramos utilizar redes mais sintéticas como é o caso do PFS/MFG que introduziremos a seguir.

3. O PFS/MFG:

PFS/MFG [Miyagi 1988] é uma rede aplicativa derivada da rede de Petri. O modelo PFS/MFG foi criado para se especificar algoritmos de controle discreto para sistemas produtivos. Estes são modelados, no PFS/MFG, em dois níveis diferentes: num primeiro nível, pelo modelo descritivo (o PFS), que é então refinado nos modelos funcionais (o MFG). O modelo resultante é, portanto, híbrido, porque este preserva a descrição das atividades, mas está detalhado no nível funcional.

3.1 Production Flow Schema (PFS):

O PFS representa conjuntamente o fluxo de materiais e de informações em um sistema produtivo, assim como os tipos de atividades e as inter-relações entre eles. O **PFS** é um grafo bipartido definido pela 3-upla **PFS=(A,D,F)**, onde **A** representa um conjunto finito de elementos ativos (atividades) **A** = $[a_i]$, **D** representa um conjunto finito de elementos passivos (distribuidores) **D** = $[d_j]$. **F** é um conjunto de arcos que representam fluxos de materiais e de informações do sistema produtivo ($\mathbf{F} = \mathbf{F_P} \cup \mathbf{F_S}$), onde $\mathbf{F_P}$ representa o fluxo principal de materiais e de informações $\mathbf{F_P} = [fp_j]$ e $\mathbf{F_S}$ representa o fluxo secundário de materiais e de informações $\mathbf{F_S} = [fs_j]$. Uma atividade corresponde a um macro-evento que representa a realização de operações elementares, tais como, manufatura, montagem, desmontagem, etc. Um distribuidor corresponde a um lugar onde fluxos entram e saem. No diagrama PFS não existem marcas, mas apenas a representação estrutural.

3.2 Mark Flow Graph (MFG):

Uma RdP do tipo C/E ao qual foram adicionadas relações de interface com o meio externo e feitas alterações visando a aplicações em sistemas de controle. O **MFG** é um grafo bipartido definido pela 6-upla $\mathbf{MFG=(B,T,A,G_I,G_E,S)}$, onde **B** representa um conjunto finito de boxes **B** = $[b_i]$, **T** representa um conjunto finito de eventos **T** = $[t_j]$. **A** é um conjunto de "arcos" de entrada e de saída das transições t_j (**A** = **(BxT)** $\cup$ **(TxB)**), onde **(BxT)** representa um conjunto de arcos de entrada para transições t_j e **(TxB)** representa um conjunto de arcos de saída de transições t_j. $\mathbf{G_I}$ é o conjunto de portas (habilitadoras e inibidoras) internas $\mathbf{G_I} = [G_{Im}]$, $\mathbf{G_E}$ é o conjunto de portas externas $\mathbf{G_E} = [G_{Em}]$ e **S** é um conjunto de arcos de saída.

O PFS/MFG é uma junção destes dois esquemas criado por [Miyagi 93] que se aplica muito bem ao design de FMS. Em uma nova formalização feita por [Silva 92] tornou-se possível executar passos elementares entre o modelo conceitual PFS e o esquema MFG de modo que é possível aplicar metodologias baseadas nesta técnica a design, validação, simulação e supervisão de sistemas com o mesmo formalismo.

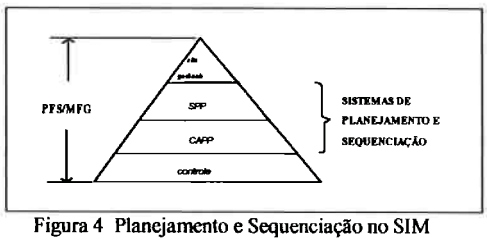


Figura 4 Planejamento e Sequenciação no SIM

3.3 Níveis de abstração do PFS/MFG:

Em controle, o PFS/MFG provê a integração total do sistema produtivo e o sistema de informações numa única arquitetura hierárquica. A Figura 4, acima, localiza os Sistemas de Planejamento e Sequenciação; SPP (Sistema de Planejamento da Produção) e PPAC (Programação da Produção Auxiliado por Computador), dentro do SIM (Sistema Integrado de Manufatura).

4. Descrição da Proposta

Propomos uma metodologia para desenvolvimento de um ambiente computacional para a automação parcial do planejamento de sistemas produtivos, baseado em IA. Usamos um mecanismo baseado na análise estrutural de propriedades da rede PFS/MFG, que constitui *o modelo conceitual* (**mc**) *do problema de "planning"*. Usamos o PFS/ MFG, direcionando os resultados para Sistemas Flexíveis de Fabricação (SFF).

4.1 As Componentes do Sistema:

A Figura 5, a seguir, ilustra as componentes do sistema proposto. O sistema de "planning" possui duas componentes: (1) o Planejador Clássico (PC) que gera o Plano Inicial (Po) e, (2) o Planejador Semi-Reativo (PSR) que tenta manter contínuamente factível um Plano Modificado (P1') a partir do Plano Corrente (P1), levando-se em conta "perturbações" ocorridas no sistema dinâmico.

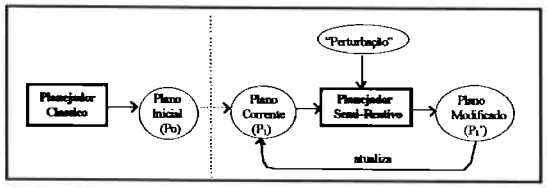


Figura 5 As Componentes do Sistema de "planning"

Este sistema não elabora planos de contingência. Ao contrário, elabora inicialmente um plano factível que é tornado o plano corrente. Então, o sistema tenta manter este plano corrente factível, atuando continuamente de uma forma semi-reativa.

O sistema está fundamentado no uso de um **modelo conceitual** (**mc**) do problema de "planning" que está baseado na teoria de Redes de Petri e da IA. Este **mc** é utilizado tanto pelo PC -- para elaborar o plano inicial (Po) -- como pelo PSR -- para manter continuamente factível um Plano Modificado (P_1') a partir do Plano Corrente (P_1), levando-se em conta "perturbações" ocorridas no sistema dinâmico. O que diferencia estas duas componentes do sistema de "planning" está na forma como cada uma delas utiliza o **mc**. PC traduz o **mc** em cláusulas PROLOG, para então aplicar um Método de "planning" Clássico para elaborar o Plano Inicial (Po). PSR traduz o **mc** em Lógica Modal Dinâmica, para então validar/modificar o Plano Corrente (P_1), elaborando (ou não) um Plano Modificado (P_1').

4.2 A Metodologia:

PASSO1: abordagem do problema como um sistema por eventos discretos, identificando os seus elementos determinantes dos possíveis estados do sistema.

PASSO2: elaboração do *modelo conceitual* (**mc**) *do problema de "planning"* do sistema produtivo, usando o PFS/MFG iniciando pelo modelo mais abstrato.

PASSO3: tradução do **mc** em cláusulas PROLOG para se aplicar o Planejador Clássico (estático).

PASSO4: construção incremental de um plano inicial (Po) usando o Planejador Clássico.

PASSO5: tradução do **mc** em Lógica Modal Dinâmica para se aplicar o Planejador Semi-Reativo (dinâmico). A axiomatização da RdP em Lógica Dinâmica está em [Silva 1992a].

PASSO6: manutenção semi-reativa do plano corrente (P1), validado ou re-planejado continuamente como um novo plano (P1'), levando-se em conta "perturbações" ocorridas dinamicamente no sistema dinâmico representado pelo **mc**, usando o Planejador Semi-Reativo Proposto.

PASSO7: validação da solução implementada fazendo testes usando exemplos cujos resultados sejam conhecidos ou possam ser verificados.

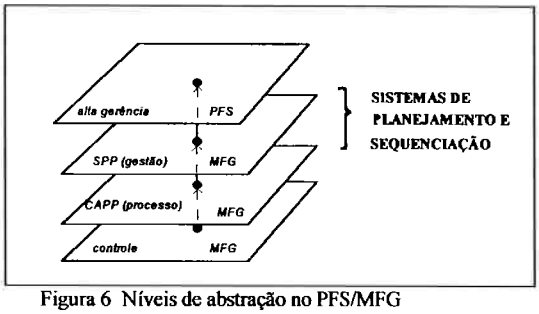


Figura 6 Níveis de abstração no PFS/MFG

5 O Modelo Conceitual (mc):

O **mc** não corresponde a um só plano mas, sim, consiste de uma representação do problema de "planning" e, como tal, corresponde à combinação de todos os planos factíveis ou não do sistema. A Fig. 6, acima, ilustra os níveis de abstração no modelo conceitual (**mc**) PFS/MFG. O modelo conceitual é híbrido sendo constituído pelo modelo descritivo(PFS) e pelos modelos funcionais(MFG). O modelo descritivo (rede PFS) para a alta gerência representa os recursos produtivos. O modelo funcional a nível de gestão ou de (SPP) é uma rede PFS/MFG que representa o plano no seu mais alto nível. O modelo funcional a nível de supervisão ou de (PPAC) é uma rede PFS/MFG que representa o plano mais detalhado que leva em conta, por ex., roteiros de produção e alocação de recursos. Uma outra aplicação do PFS/MFG reunindo SPP, CAPP e o nível de controle foi proposto em [Kagoha 94] explorando as características de

refinamento da nova formulação do PFS/MFG [Silva 94].

5.1 O Planejamento Semi-Reativo (PSR):

O Planejador Semi-Reativo (PSR) baseia-se em Provador Automático de Teoremas (PAT) para a Lógica Modal. [Bibel 1983] apresenta um PAT para Lógica de Predicados de Primeira Ordem (LPPO), onde se situa a Lógica Modal. PSR faz uma análise da estrutura da RdP. Por exemplo, identifica, a partir da estrutura da RdP, de possíveis formas para direcionar o processo de busca de soluções. Verifica propriedades comportamentais, como vivacidade, conservatividade, caracterização de processos paralelos e em conflito, sincronismo (distância síncrona), etc. Associa regras para escolha, p. ex., de operações com conflito no uso de recursos.

Em [Kagoha 94] foi proposto um sistema onde o planejamento da produção, feito a nível de SPP é propagado para o nível de design (PPAC) e posteriormente para o nível fábrica e suas estações de controle. Neste caso o que chamamos de "layer" fica bastante claro, e está associado a estes três níveis básicos. Evidentemente, em um sistema flexível, é possível que dois ou mais produtos sejam manufaturados ao simultaneamente e, neste caso, a passagem do nível design para o nível fábrica só se completa quando o design de ambos é finalizado. Se este processo não é sincronizado, a passagem ainda pode ser representado na nova formulação do PFS/MFG, mas, ainda assim, tomamos a finalização do design de todos os produtos do mix como um marco da transferência de controle do nível design para o nível fábrica.

No caso de alguma ação emergencial (devido a erros ou situações externas inesperadas durante o processo de manufatura), os níveis mais próximos do chão de fábrica têm precedência sobre os mais abstratos. O sistema é Semi-Reativo porque trata apenas dos caos em que ocorre um acréscimo de novas tarefas. O cancelamento de tarefas previstas é tratado simplesmente eliminando-se a sua execução do plano corrente. O sistema considera apenas casos onde as modificações necessárias no plano original possam ser consideradas como *perturbação*, isto é, não causam mudanças muito drásticas no plano. Deste modo, cada "layer" pode desenvolver, usando técnicas de IA, um plano mutante, baseado no conhecimento adquirido na análise da estrutura do problema, que manterá o sistema funcionando em padrões baixos de otimização até que um novo plano o substitua (Fig. 7).

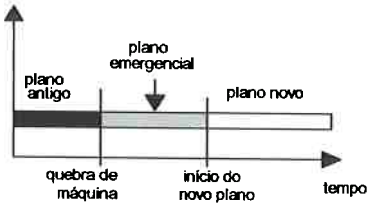


Fig. 7 Intervalo coberto pelo "plano emergencial"

A metodologia proposta contempla ainda restrições de tempo, considerado na representação do PFS/MFG que estamos usando como um parâmetro associado às atividades [Silva 92]. Conhecendo as estimativas de tempo associadas a cada atividade do processo, e tendo uma boa estimativa do tempo de mutação em cada "layer" é possível avaliar o tempo em que o plano emergencial vai estar ativo. Apesar de ser uma avaliação heurística e não uma análise qualitativa, esperamos que isto nos forneça base suficiente para minimizar o tempo de vida do plano emergencial, pelo menos no que depender do design do sistema.

6. Uma aplicação em SFF

Como uma aplicação mais realística das idéias centrais relacionadas com o nosso trabalho, apresentamos a sequenciação de tarefas num SFF do tipo "job shop" [Shimad 93], com quatro tipo de peças e com três máquinas. Cada peça apresenta dois roteiros de produção; como mostra a Tabela 1, a seguir.

Peça		P1	P2	P3	P4
Tarefas		O1 O2 O3	O4 O5 O6	O7 O8 O9 O10	O11 O12
Pre-C		- O1 O2	- - O4,O5	- - - O9	- -
Proces	Máq.	M2 M1 M3	M1 M2 M3	M3 M1 M3 M1	M3 M2
Básico	Tempo	4 5 2	6 3 3	3 3 6 2	4 3
Proces	Máq.	M3 M2 M1	M2 M3 M1	M3 M3 M2 M3	M2 M1
Altern.	Tempo	6 6 4	6 4 5	4 5 7 2	4 5

Tabela 1 Processos para Manufatura em SFF

Os passos da metodologia, no item 4.1, são:

PASSO1: Os elementos fundamentais para o planejamento do sistema dinâmico são: horizonte de planejamento (τ), demanda, linha de produtos (P1, P2, P3 e P4), recursos produtivos (M1, M2 e M3), roteiro de produção e factibilidade do plano.

PASSO2: . Vamos modelar o problema usando PFS/MFG. A Fig.8, a seguir, mostra o diagrama PFS do sistema.

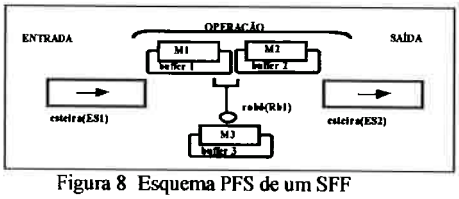


Figura 8 Esquema PFS de um SFF

A Fig.9, a seguir, mostra a rede PFS/MFG que representa o plano de mais alto nível, no nível de gestão ou de SPP. Admite-se o uso da

política de estoque "just-in-time", i.e., não há nenhum estoque de insumos ou produtos

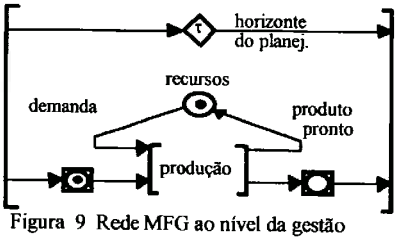


Figura 9 Rede MFG ao nível da gestão

Este plano deve ser detalhado para o nível de supervisão para atender as restrições tecnológicas: roteiros de produção e alocação de recursos compartilhados.

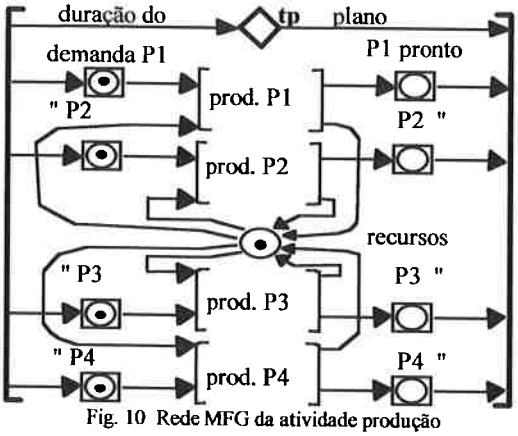


Fig. 10 Rede MFG da atividade produção

A Fig.10, acima, mostra a rede MFG para a atividade de produção da peça **P1**. A Figura 11, a seguir, mostra a rede MFG para a atividade de produção da peça P1.

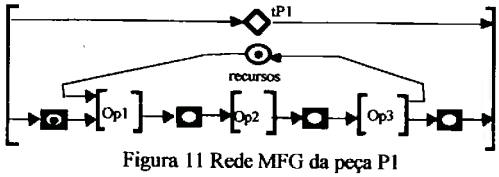


Figura 11 Rede MFG da peça P1

A duração plano **tp** = max(tPi, i=1,,4). Para que o plano seja factível, a duração do plano **tp** deve ser menor que o horizonte do planejamento τ. A Fig. 12a mostra a rede PFS/MFG para os roteiros de produção para a operação Op1 na peça P1.

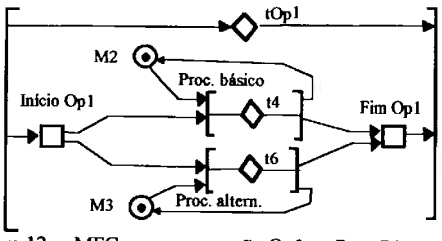


Figura 12a - MFG para a operação Op1 na Peça P1

A duração da Op1 depende tOp1 depende do roteiro escolhido. Tem-se, deste modo, um estimador grosseiro para a **tp**. Esta é feita a nível de gestão. Se o horizonte do planejamento τ for maior que **tp** (pessimista e/ou otimista) estimada, o plano poderá ser detalhado no nível mais abaixo.

PASSO3: Usamos a seguinte regra de prioridade para escolher a operação seguinte quando há conflito o uso de recursos: o menor tempo de processamento.

PASSO4: simulação da rede: Se a demanda é unitária para cada produto e todas as máquinas disponíveis. Então, para τ = 24 horas, o estimador obtém os seguintes limitantes:

a) **tp** otimista = max(16,10,18) = 18 e,

b) **tp** pessimista = max(14,23,21) = 23.

A condição necessária está atendida, τ > tp. Portanto, continuamos ao nível de supervisão.

PASSO5: implementamos este problema em computador usando o ARITY/PROLOG.

PASSO6: Usando o programa acima, obtivemos a alocação mostrada na Figura 13, a seguir.

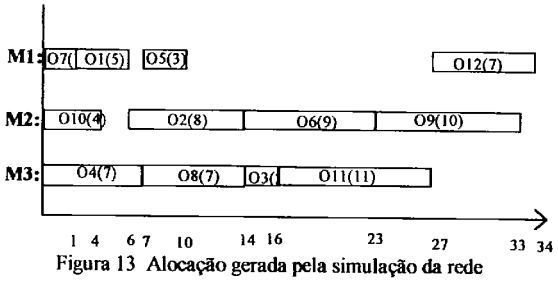


Figura 13 Alocação gerada pela simulação da rede

7. Conclusão

A sequenciação gerada na aplicação descrita no item 6, acima foi encontrada dentro de um tempo de processamento de 5 segundos, usando um microcomputador tipo IBM PC 486 DX2 de 66 Mhz. O resultado da implementação referida no item 6, acima, mostrou ser uma solução eficiente e sem "backtracking" na falha para problemas de planejamento em sistemas dinâmicos, i.e., com qualidade assegurada desde o princípio.

Estamos no momento trabalhando no desenvolvimento de um sistema inteligente para a geração de planos mutantes. Este sistema estará acoplado com um ambiente de RdP de modo a obter de forma automática os dados da análise estrutural do problema de planejamento. Uma integração completa destes dois módulos em um ambiente único está fora do escopo deste trabalho.

*****

Bibliografia

[Bibel 1983] BIBEL, w. Matings in Matrices. Comm. of ACM. Vol 26, n. 11. 1983. pp. 844-852.

[Drummond 1985] IJCAI 85 (also in Readings on Planning 1990).

[Fikes et Nilsson 1971], 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.

[Hendler 1992], HENDLER, J. et SPECTOR, L. Planning and control across supervenient levels of representation. Department of Computer Science. University of Maryland. Third Draft: sept 1992, for International Journal on Intelligent & Cooperative Information Systems (IJICIS).

[Kagoha 1994] KAGOHARA, M., TOLEDO, C., SILVA, J.R., MIYAGI, P.E.; Automatic Generation of Control Programs for Manufacturing Cells. In:: IFIP WG5.7 WORKING CONFERENCE ON EVALUATION OF PRODUCTION MANAGEMENT METHODS, Gramado, 1994. Anais:. 1994.

[Monner 1994] MONNERAT, N., SILVA, J.R., Analise de Propriedades Estruturais das Redes de Petri. BOLETIM TÉCNICO DA EPUSP/PMC, São Paulo, to appear.

[Miyagi 1993], MIYAGI, P.E. Controle de Sistemas de Eventos Discretos: Aplicação em Automação Industrial. SÃo Paulo, SP. 1883. Tese (livre docência). UPUSP.

[Santos Fo. 93] SANTOS FILHO, Diolino José dos, Realização do Controle de Tarefas de Veículos Autônomos de Transporte Através do MFG Estendido com Marcas Individuais. In: CONG-RESSO NACIONAL DE AUTOMAÇÃO, São Paulo, 1994, Anais : CONAI '94.

[Sacerdoti 1974], SACERDOTI, E.D. Planning in a hierarchy of abstraction spaces. Artificial Intelligence 5, 1974. ALLEN, J.; HENDLER, J.; TATE, A., coord. **Readings in planning**. San Mateo, California. Morgan Kaufmann, 1990. Cap.2, p.98-108.

[Shimad 1994] SHIMADA, L.M.; Uma Metodologia baseada em Redes de Petri para o Planejamento Inteligente de Sistemas. EPUSP, 1994, Qualificação para Doutorado em Eng. Mecânica.

[Silva 1992] SILVA, J.R.; PESSOA, F. J. B. Análise Semi-Automática de Mark Flow Graphs. In: WORKSHOP IBERO-AMERICANO DE SISTEMAS AUTONO-MOS EM ROBOTICA E CIM, Lisboa, 2/4 de novembro de 1992. Revista de Informaçâo Técnica e Científica "Robotica e Automatizaçâo", Fasciculo 10, p. 25 a 30, nov. 1992, Lisboa. Ed. Publindústria Produçâo de Comunicaçâo Lda

[Silva 1992a] SILVA, J.R. Uma formalização do processo formal baseado em metáforas: sua aplicação na automatização de sistemas de eventos discretos. São Paulo 1992. Tese (Doutorado). Escola Politécnica. Universidade de São Paulo.

[Silva 1994] SILVA, J.R.; COWAN, D.D.; LUCENA, C.J.P.; The Design Reusability Problem; A Case-Based Approach to the Design of Flexible Manufacturing Systems. Relatório Técnico USP, 1994.

[Sussman 1974], 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.

[Tate 1977], TATE, A. Generating project networks. IJCAI 1977. In: ALLEN, J.; HENDLER, J.; TATE, A., coord. **Readings in planning**. San Mateo, California. Morgan Kaufmann, 1990. Cap.5, p.291-296.

TATE, A., coord. **Readings in planning**. San Mateo, California. Morgan Kaufmann, 1990. Cap.3, pp. 118-139.

Agradecimentos

CNPq, FAPESP, Flexsys(Esprit 76101),AP3I.

[i] Trabalho apresentado no 2° Simpósio Brasileiro de Automação Inteligente. 13 a 15.set.1995. CEFET-PR. Curitiba.