UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

Uma Heurística para a Programação da Produção de Sistemas Flexíveis de Manufatura usando Modelagem em Redes de Petri

EDUARDO GOMES RIBEIRO MAGGIO

Dissertação de Mestrado apresentada ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: **PROF. DR. ORIDES MORANDIN JUNIOR**

São Carlos - SP
Maio de 2005

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

M188uh

Maggio, Eduardo Gomes Ribeiro.
Uma heurística para a programação da produção de sistemas flexíveis de manufatura usando modelagem em redes de Petri / Eduardo Gomes Ribeiro Maggio. -- São Carlos : UFSCar, 2007.
107 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Programação da produção. 2. Busca heurística. 3. Sistemas flexíveis de manufatura. 4. FMS. 5. Redes de Petri virtuais. I. Título.

CDD: 658.53 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Uma Heurística para a Programação da Produção de Sistemas Flexíveis de Manufatura Usando Modelagem em Redes de Petri"

EDUARDO GOMES RIBEIRO MAGGIO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Orides Morandin Junior
(Orientador – DC/UFSCar)

Prof. Dr. Estevam Rafael Hruschka Jr
(DC/UFSCar)

Prof. Dr. Edílson Reis Rodrigues Kato
(Uniara)

Prof. Dr. Paulo Eigi Miyagi
(POLI/USP)

São Carlos
Maio/2005

Dedicatória

À memória de meus avôs
Luiz Gomes Ribeiro e
Jacintho Maggio

À memória de meu tio
Sebastião Maggio

Agradecimentos

À minha companheira de longa data, Flávia, pelo apoio e compreensão no período em que me empenhei neste trabalho, me privando, muitas vezes, de seu convívio.

Aos meus pais, Euripes e Áurea, pelo suporte, pelos conselhos e por sempre acreditarem em mim.

Ao meu irmão, Renato, sempre um exemplo de força de vontade, determinação e presença de espírito, a quem eu admiro e procuro me espelhar.

Ao orientador e amigo, Prof. Dr. Orides Morandin Jr., que me mostrou o "caminho das pedras" no árduo percurso da elaboração deste trabalho, sempre com companheirismo e paciência.

Ao Allan, pela amizade, pelo companheirismo e pelos trabalhos que fizemos juntos.

Ao Ricardo, pelo seu exemplo de persistência e de como encarar as adversidades com calma e serenidade.

Ao amigo Escovar, pelos seus conselhos que foram de grande ajuda.

Aos professores do programa de mestrado de que fui aluno, Profa. Dra. Maria do Carmo Nicoletti, Prof. Dr. Hélio Crestana Gaurdia, Profa. Dra. Heloísa de Arruda Camargo, Prof. Dr. José Hiroki Saito.

Ao Prof. Edílson Kato, pela sua imensa colaboração.

A todos que de alguma maneira contribuíram para a elaboração deste trabalho.

Sobretudo agradeço a Deus.

Resumo

MAGGIO, Eduardo G. R. *Uma Heurística para a Programação da Produção de Sistemas Flexíveis de Manufatura Usando Modelagem em Redes de Petri.* Dissertação de Mestrado; Departamento de Computação, Universidade Federal de São Carlos; Maio de 2005; 119 p.

Abordagens de Busca baseadas em Rede de Petri (PN) têm sido mostradas como uma forma promissora de resolver o problema da Programação da Produção de Sistemas Flexíveis de Manufatura (FMS). Entretanto, o tempo de resposta é crítico, uma vez que se trata de um sistema de alta complexidade computacional. Focando a redução do tempo de resposta do sistema, este trabalho propõe uma heurística para busca baseada em Rede de Petri para resolver o problema de programação de FMS na minimização do *makespan*. Experimentos mostraram um avanço na melhoria do tempo de resposta em relação a trabalhos anteriores.

Palavras-chave: Programação da Produção, Sistemas Flexíveis de Manufatura, FMS, Busca Heurística, Rede de Petri Virtual.

Abstract

MAGGIO, Eduardo G. R. *A Heuristic for Scheduling of Flexible Manufacturin System modeled with Petri nets.* Master's Degree Dissertation. Department of Computer Science, Federal University of São Carlos; May, 2005; 119 p.

The Petri Net based Search has been shown as a promising way to solve Flexible manufacturing Systems (FMS) Scheduling Problem. However, the response time is critical since it's a system with high computational complexity. Focusing the reduction of response time, this work proposes a heuristic for Petri Net based Search to solve FMS Scheduling problem of *makespan* minimization. Experiments showed improvements on response time reduction comparing with prior works.

Keywords: Scheduling, Flexible Manufacturing System, FMS, Heuristic Search, Virtual Petri Net.

Sumário

Lista Símbolos

(termos agrupados segundo o contexto do uso)

Pesquisa Operacional

f	tempo de percurso (flowtime)
f_{MAX}	tempo de percurso máximo
f_{me}	tempo de percurso médio
f_i^*	mínimo tempo de percurso (*flowtime ideal*)
$\phi_i(n)$	mínimo tempo de percurso remanescente a partir do estado n

Redes de Petri

t	transição
p	lugar (*place*)
w	peso do arco
M	marcação de rede de Petri
$M(p)$	número de *tokens* em uma *lugar p* para uma dada marcação M em uma rede de Petri
M_0	marcação inicial
P	conjunto de lugares
T	conjunto de transições
F	conjunto de arcos $F \subseteq (P\times T) \cup (T\times P)$
$w(\cdot,\cdot)$	função de ponderação dos arcos W: $F\rightarrow\aleph$
p_i	i-ésimo lugar em uma rede de Petri
t_i	i-ésima transição em uma rede de Petri
N	estrutura de rede de Petri
$L(N, M_0)$ $L(M_0)$	conjunto de todas as possíveis seqüências de disparo em uma rede (N, M_0)
$R(N, M_0)$ $R(M_0)$	conjunto de todos as marcações atingíveis a partir de M_0 em uma rede (N, M_0)
$I(\cdot,\cdot)$	função de arco de entrada: $(P\times T)\Rightarrow\aleph$
$O(\cdot,\cdot)$	função de arco de saída: $(T\times P)\Rightarrow\aleph$
$C_{m\times n}$	matriz de incidência
I_μ	matriz identidade de ordem μ
$\boldsymbol{M}$	marcação temporizada
$Ct(\boldsymbol{M})$	tempo corrente da marcação temporizada $\boldsymbol{M}$
$k(p)$	capacidade de um lugar p
σ	seqüência de disparos de transição
u	*token* de PN de alto nível (ex.: CPN)
U	conjunto de pares ordenados associados a *tokens*
p_u	lugar da PN em que um *token u* se encontra
$\tau(u)$	tempo de reminiscência (ou tempo remanescente) do *token u*

Aspectos Formais de Computação

$\boldsymbol{G}$	grafo
$\boldsymbol{V}$	conjunto de vértices
$\boldsymbol{E}$	conjunto de arcos rotulados
$\boldsymbol{P}$	conjunto de regras de produção de uma gramática formal
$\boldsymbol{T}$	grafo em árvore
Gram	gramática formal

Algoritmos de Busca e Funções Heurísticas

n	nó da árvore de busca
Ht^*	mínimo custo total
$Ht(n)$	real custo mínimo remanescente a partir de n
$f(n)$	função para a estimativa do custo total de percurso envolvendo o nó de busca n
$h(n)$	função heurística (estimativa do custo mínimo remanescente a partir de n)
$g(n)$	custo de percurso no alcance do nó n
q	estado de busca
q_0	estado inicial
Υ	conjunto de ações
y	ação (operador sobre estados do problema)
$c(q, y, q')$	custo de passo de um estado q para q' pela ação y

Complexidade de Problemas

$\mathbf{O}(\cdot)$	notação para análise assintótica
b	fator de ramificação
d	profundidade do nó mais raso
l	limite de profundidade para restrição da busca
m	comprimento do maior dos caminhos em um espaço de estados

Notação específica

Rt	número total de máquinas no sistema
$Rrs(n)$	recursos ainda necessários para a finalização do produto de maior *flowtime ideal remanescente* no estado n
$Rr_i(n)$	número de máquinas pelas quais um certo produto em processo i, estando no estado n, ainda precisa passar
Q	conjunto dos lugares que representam *buffers*
$j(n)$	função de custo
$a, b, c, d, e, f, g, h, z$	letras estilizadas correspondendo a símbolos terminais de uma linguagem formal para representarem operações no chão de fábrica de um FMS

$\aleph$	conjunto dos Números Naturais

Alguns símbolos não constam desta lista, ou por serem objeto de outros trabalhos devidamente referenciados, ou por estarem próximos a sua definição. Consideram-se comumente empregados os demais termos ausentes nesta lista.

Lista de Figuras

Siglas

Sigla	***Significado***	***Consultar***
AGV	Veículo Auto Guiado	Item 2.3 do capítulo 2 (nota de rodapé)
CIM	Manufatura Integrada por Computador	Item 2.2 do capítulo 2
CNC	Máquinas de Controle Numérico por Computador	Item 2.2 do capítulo 2
CPN	Rede de Petri Colorida	Item 3.4.1 do capítulo 3
DLSS	Dynamic Look-ahead Stage Search	Capítulo 5
DWS	Dynamic Windows Search	Item 4.5.2 do capítulo 4
FMS	Sistema Flexível de Manufatura	Item 2.3 do capítulo 2
GA	Algoritmo Genético	Item 4.6.3 do capítulo 4
HLPN	Rede de Petri de Alto Nível	Item 3.4.1 do capítulo 3
IGS	Inteligent Generator of Successors	Capítulo 5
LIAA	Laboratório de Inteligência Artificial e Automação	Item 7.1 do capítulo 7
PCP	Programação e Controle da Produção	Item 2.4 do capítulo 2
PMP	Programa-Mestre da Produção	Item 2.3 do capítulo 2
PN	Rede de Petri	Capítulo 3
RCR	matriz de alcançabilidade de custo de recurso	Capítulo 5
S^4R	Systems of Sequential Systems with Shared Resources	Capítulo 5
SMA*	Simplified Memory-bounded A*	Item 4.5.2 do capítulo 4
TPN	Rede de Petri Temporizada	Item 3.4.2 do capítulo 3
VPN	Rede de Petri Virtual	Item 3.4.3 do capítulo 3
WIP	Work-In-Progress (entidades em curso)	Item 2.5 do capítulo 2

CAPÍTULO 1 *Introdução*

A fim de obter vantagem competitiva e sobreviver com sucesso no atual ambiente empresarial globalizado, a indústria tem incorporado técnicas de automação visando suprir a necessidade de uma rápida adaptação às mudanças no sistema de produção, garantindo aceitável volume de produção e variabilidade de produtos.

Neste contexto, o Sistema Flexível de Manufatura (FMS) vem sendo apresentado como uma opção de tecnologia na automação dos processos condizente com as características de produção de significativa parcela do mercado.

Neste tipo de sistema, a versatilidade é obtida em diferentes aspectos, possibilitando uma série de benefícios para as empresas. No entanto, toda esta flexibilidade implica novos desafios no âmbito da programação da produção.

A programação da produção é uma das mais complexas atividades que compõem o planejamento e controle da produção (PCP), que é a área responsável pelas decisões do emprego dos recursos de produção de forma a assegurar a adequada concretização dos objetivos previstos em uma organização. No nível de decisão de PCP referente à programação da produção, busca-se obter a melhor agenda de execuções de tarefas no processo de transformação de insumos do sistema, de forma a atingir certos critérios de desempenho.

Um ponto dentre as dificuldades desta atividade encontra-se, já de início, na formulação e na modelagem do problema da programação de um FMS, que devem ser feitas de forma a obter um modelo que represente com fidelidade as principais características de um FMS.

Várias pesquisas apontam o uso de rede de Petri (PN) como uma forma conveniente para modelar as características de um FMS e que, além disso, fornecem um respaldo analítico que permite sua aplicação em conjunto com um número considerável de estratégias possíveis para a resolução do problema da programação da produção.

Diferentes considerações na definição do problema foram assumidas em trabalhos da área. Igualmente, no tocante às PN, formas de modelagem de FMSs vêm sendo investigadas, incluído aplicações em outros contextos, mas que podem ser usadas para tratamento do

problema em questão. Além disso, várias formas de solução vêm sendo propostas para o uso em conjunto com rede de Petri na resolução da programação da produção em um FMS, como o uso de estratégias de busca por exploração dos possíveis estados do sistema.

Entretanto, neste enfoque, a questão da complexidade computacional, mediante as inúmeras possibilidades a serem consideradas, inviabiliza a obtenção da programação da produção mais adequada em tempo viável, segundo um determinado parâmetro.

Como será visto no capítulo 5, vários estudos vem sendo desenvolvidos ao longo dos anos na tentativa de se desenvolver sistemas que encontrem uma solução que se aproxime da ideal, segundo critérios objetivos, em um tempo reduzido.

Diferentes formas de reduzir o tempo de resposta vêm sendo exploradas para o tipo de abordagem em questão. Sugestões foram feitas ora no que se refere às características que devem ser modeladas no sistema ora na forma de se encontrar a solução sobre um dado modelo. Neste segundo aspecto, têm-se desenvolvido estratégias buscas e funções heurísticas na tentativa de se obter certa eficiência tanto na rapidez com que uma resposta é obtida como na proximidade que a solução encontrada apresenta da que se supõe ser a ideal. Porém, o problema da viabilidade do tempo de resposta ainda perdura.

Neste contexto, este trabalho propõe uma função heurística focando a redução do tempo de obtenção de uma programação de sistemas flexíveis de manufatura em uma abordagem baseada na modelagem em redes de Petri e em estratégia de busca por exploração de estados do sistema modelado.

Para tanto, é apresentada nos capítulos subseqüentes uma introdução aos conceitos envolvidos e a proposta. No capítulo 2, delineia-se uma visão geral sobre programação da produção em um FMS no contexto do planejamento e controle em sistemas de produção.

É apresentada no capítulo 3 uma breve revisão sobre rede de Petri, definindo seus conceitos básicos, além de apresentar suas principais propriedades e variações.

Na seqüência, no capítulo 4 é reunido um número de estratégias de busca para solução de problemas, discutindo aspectos referentes aos algoritmos relacionados e a idéia básica de funcionamento dos mesmos.

Então, no capítulo 5, mostra-se uma breve revisão sobre as aplicações que vêm sendo feitas na programação de FMSs envolvendo modelagem em rede de Petri.

Com o uso das exposições feitas nos capítulos anteriores mostra-se, no capítulo 6, a heurística proposta.

No capítulo 7, expõem-se os experimentos realizados, comparando o desempenho da heurística proposta em relação às de trabalhos anteriores.

Conclusões e considerações para trabalhos futuros são apresentadas no capítulo 8.

CAPÍTULO 2 *Programação de Sistemas Flexíveis de Manufatura*

SEGUNDO Slack *et al*. (1999), a atividade de programação, também conhecida como *scheduling*, é uma das mais complexas tarefas no planejamento e controle da produção. A automatização do processo de transformação de materiais na produção pode ser feita pelo uso de Sistemas Flexíveis de Manufatura (FMS) que proporcionam certa versatilidade adequada a certos tipos de organização, mas implica também um desafio na obtenção de uma melhor programação.

Esse capítulo apresenta uma visão geral sobre esses aspectos partindo de um contexto mais amplo. Para isso, são introduzidos conceitos de sistemas de produção e tecnologias de manufatura identificando o ponto em que FMS se insere, tratando-o em detalhes adiante. Em seguida, é apresentada uma visão de programação da produção e, então, sua aplicação em FMS, buscando uma melhor definição do problema.

2.1 Sistemas de Produção

O sistema de produção está no núcleo de uma organização industrial, pois trata uma competência fundamental desta – a produção. Outros papéis fundamentais de uma organização industrial são o *marketing*, o controle contábil-financeiro e o desenvolvimento do produto e/ou serviço, exercidos junto às funções de apoio (recursos humanos, compras, engenharia e suporte técnico). A produção pode ser entendida, de forma ampla, como o processo de transformação envolvendo recursos de entrada (pessoal, maquinário e/ou insumos em geral) para geração de resultados como bens e serviços (Slack *et al.*, 1999). No escopo deste trabalho, o enfoque está nas *operações de manufatura*, em que os materiais são processados em um chão de fábrica por máquinas ou operadores para a geração de produtos.

Dentre as dimensões sobre as quais a produção pode ser classificada, pode-se destacar o volume de produção e variedade de tipos de produtos. A posição da produção no espectro de volume-variedade influencia muitos aspectos de sua *atividade de projeto*, e determina a forma como os seus critérios de desempenho (qualidade, rapidez, confiabilidade, flexibilidade e custo) são definidos. Apresenta-se na figura 2.1 os tipos de processo em operações de manufatura nesse espectro, descritos em maiores detalhes a seguir.

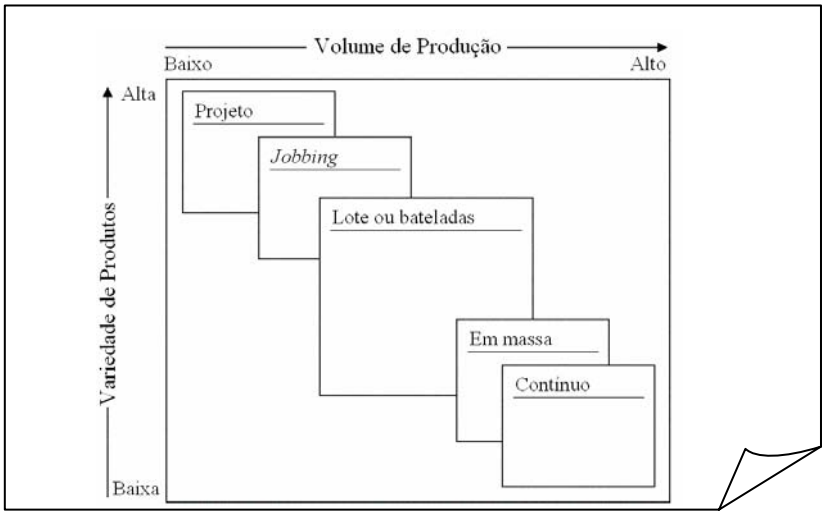

FIGURA 2.1 Tipos de processo em operações de manufatura (adaptado de Slack et al., 1999).

Processos de projeto envolvem a produção de produtos com alto grau de especificidade com baixo volume e alta variabilidade, sendo o tempo de produção relativamente longo, como, por exemplo, a instalação de um sistema de computadores, a construção de navios, produção de filmes e a maioria das atividades das companhias de construção.

Processos de *jobbing* também envolvem produtos de alta variabilidade e baixo volume. Porém, nos processos de projeto, os recursos são dedicados com certo grau de exclusividade. Já, nos processos de *jobbing*, há um maior compartilhamento de recursos para a produção de diferentes tipos de produto. Esses processos apresentam mais itens mas usualmente em número menores que os de projeto, ainda oferecendo baixo grau de repetição.

Processos em lote ou bateladas são similares aos de *jobbing*, apresentando produtos com menor grau de variedade e sendo a produção realizada segundo uma quantia definida – o *tamanho do lote*. Dependendo desse tamanho e da facilidade de adaptação do tipo de produto à operação, pode-se obter razoável grau de repetição.

Processos de produção em massa são os que produzem em larga escala, apresentando produtos de baixa variabilidade. As diversas variantes do produto não afetam o processo básico de produção. Uma fábrica de automóveis pode estar neste contexto, se mesmo com variações específicas (tamanho do motor, cor, equipamentos extra) as atividades forem essencialmente as mesmas e amplamente previsíveis.

Processos contínuos operam em volumes ainda maiores que os de produção em massa e, em geral, com produtos de menor variabilidade. O aspecto contínuo pode estar relacionado ao fluxo ininterrupto de materiais ou pelo fato da operação suprir produtos sem uma parada. Geralmente estão associadas a tecnologias de comportamento mais rígido, de capital intensivo com fluxo de materiais altamente previsível. Tem-se como exemplo de processos contínuos as refinarias petroquímicas e indústrias siderúrgicas.

O tipo de processo também pode influenciar o tipo de arranjo físico dos recursos de transformação e a forma como lidar com o fluxo de materiais entre estes. Os quatro tipos básicos de arranjo físico são:

- *Posicional* – normalmente é usado quando os recursos transformados são, ou muito grandes, ou muito delicados, ou difíceis de serem movidos.
- *Por processo* – em que os recursos similares de operações são mantidos juntos e os produtos que sofrem as transformações percorrem seu roteiro de acordo com suas necessidades de processamento. Em geral, é usado quando a variedade de produtos é relativamente alta.
- *Celular* – em que os recursos necessários para uma classe particular de produtos são agrupados de alguma forma.
- *Por produto* – é aquele em que os recursos de transformação estão configurados na seqüência específica para melhor conveniência do processo de obtenção de um certo produto ou tipo de produto.

A posição da produção no espectro volume-variedade influencia seu arranjo físico e, conseqüentemente, o fluxo dos recursos transformados como é mostrado na figura 2.2, estando na ordem crescente do nível de volume e decrescente no grau de variabilidade a seqüência de arranjos físicos: posicional, por processo, celular e por produto.

2.2 Tecnologias de Manufatura

As tecnologias de manufatura são aquelas envolvidas na atividade de transformação de materiais nesse tipo de sistema de produção. Elas podem ser, também, classificadas de acordo com o nível de volume de produção e o grau de variedade dos produtos, como é mostrado na figura 2.3, e são descritas a seguir:

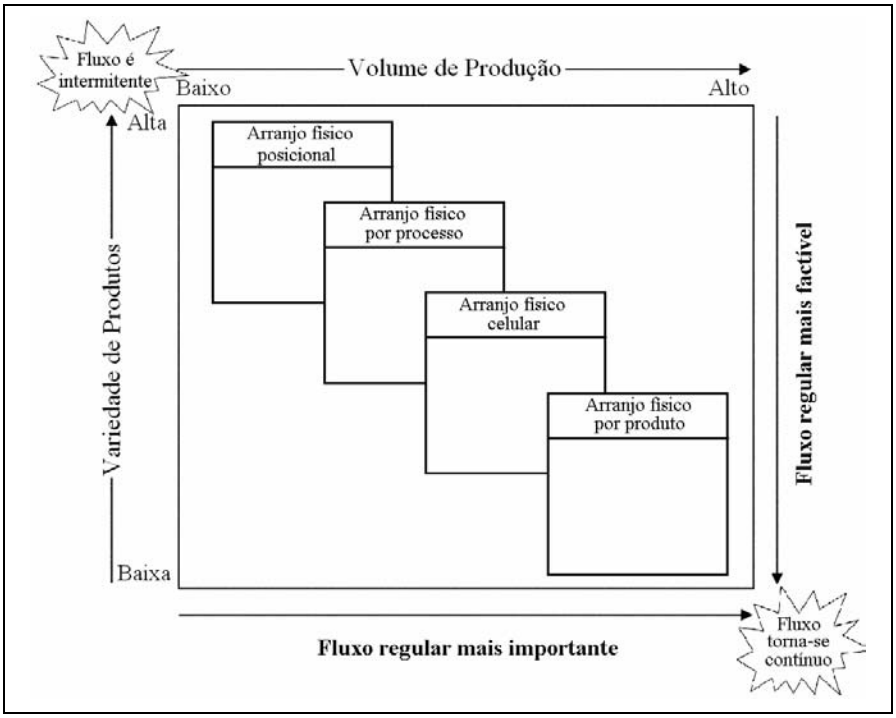

FIGURA 2.2 Influência do volume e variabilidade no arranjo físico (adaptado de Slack et al., 1999)

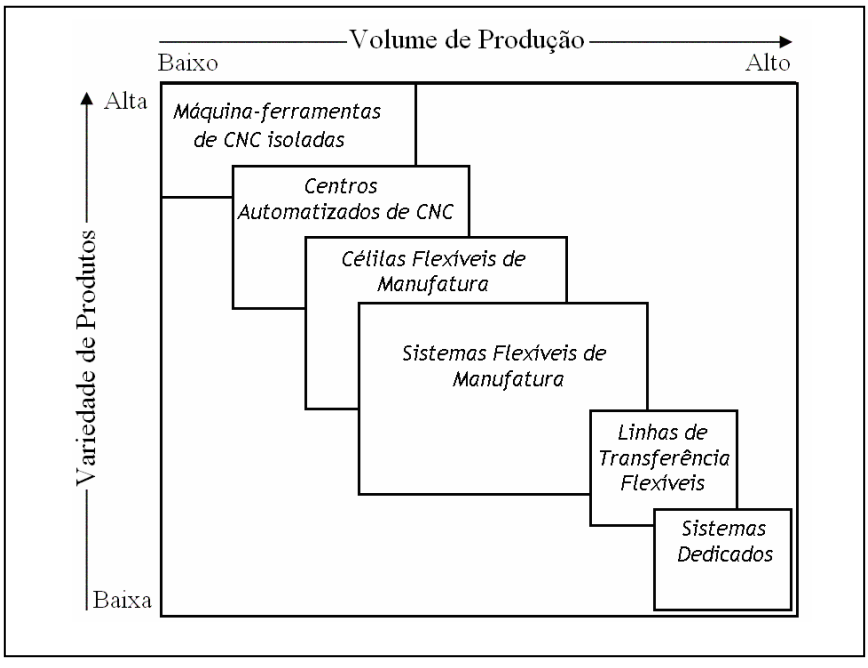

FIGURA 2.3 As características de volume-variedade das tecnologias de produção (Slack *et al.*,1999).

- As *máquinas-ferramentas de comando numérico por computador* (CNC) possuem um sistema de controle que identifica instruções armazenadas em um

computador dedicado e as convertem em operações de máquinas, controlando o movimento das ferramentas e a velocidade da operação ao longo do processamento.

- Os *centros automatizados de comando numérico por computador* são compostos de máquinas-ferramentas CNC que possuem um magazine de ferramentas com trocas automáticas, que torna flexível as trocas de operações de tipos diversos. Também, em comparação com as máquinas CNC, apresentam maior flexibilidade no movimento de peças comuns definido em termos de *grau de liberdade de movimento* da máquina.

- As *células flexíveis de manufatura* caracterizam-se por conter agrupamentos de máquinas, em sua distribuição física de maquinário, chamados de *células.* A transferência de materiais entre máquinas de *células* diferentes, em geral, não é feita de forma automática.

- Os *sistemas flexíveis de manufatura* (FMSs) são compostos por estações de processamento interligadas por um sistema de transporte de materiais e armazenagem temporária, sendo todo o processo coordenado por uma estação central de processamento. A flexibilidade deste tipo de sistema se deve, entre outros aspectos, à capacidade de se produzir, simultaneamente, diferentes tipos de peças nas várias unidades de trabalho (Groover, 1987).

- Nas *linhas de transferência flexíveis*, ligando as máquinas de processamento tem-se um sistema automatizado de transporte em linha. A flexibilidade está na capacidade de produzir tipos diferentes de produtos.

- Os sistemas dedicados, como sugere o nome, são projetados para produzir um único tipo de produto, com variações praticamente inexistentes e um alto volume de produção, o maior neste espectro.

A crescente integração de tecnologias de manufatura traz o conceito de manufatura integrada por computador (CIM), dentro do qual o FMS está inserido.

2.3 Sistema Flexível de Manufatura (FMS)

Um *sistema flexível de manufatura* (FMS) é um tipo de sistema de manufatura integrada por computador que consiste em um grupo de estações de processamento (predominantemente máquinas-ferramentas CNC), em que há um sistema de transporte de material e de armazenagem local temporária. O sistema de controle é distribuído em computadores dedicados às estações, e tudo, incluindo o tratamento de material, é coordenado por uma estação de central de controle.

O uso de FMS traz as vantagens da obtenção de o grau de variedade de produto desejado, dependendo de seu arranjo físico, e um nível de volume de produção razoavelmente grande. Um exemplo de FMS é esquematizado na figura 2.4, com quatro estações de processamento e um "trilho" indicando a topologia do sistema de transporte de peças por AGV[1] (veículos auto-guiados). Cada estação, no caso, possui terminais de trilhos em suas laterais e centro, permitindo a carga e a descarga de materiais na área de processamento, no interior da estação, e em seus respectivos *buffers* de entrada e saída.

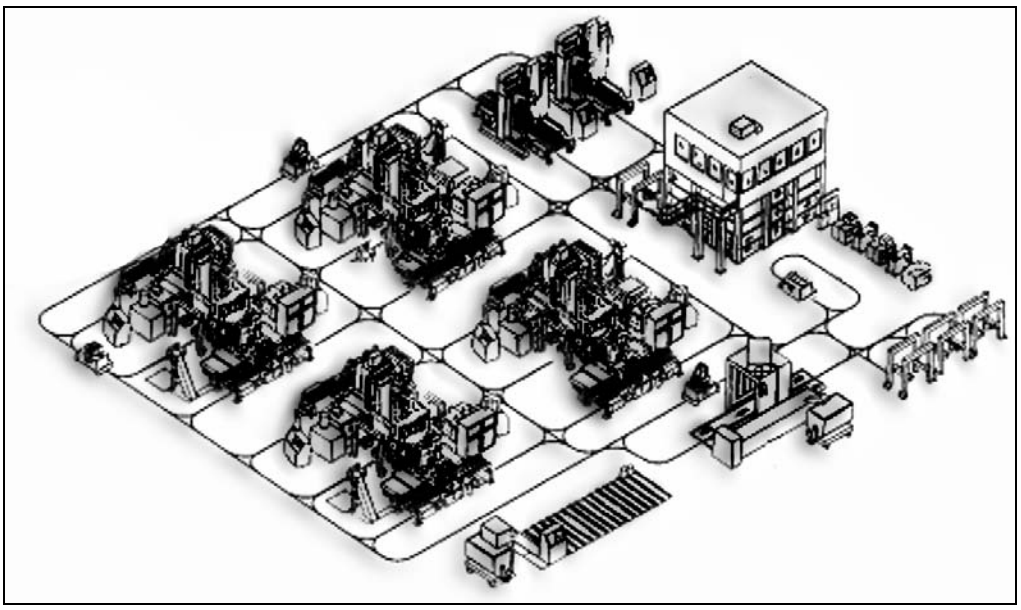

FIGURA 2.4 Exemplo de um FMS, adaptado de Groover, 1987.

[1] AGV vem do termo inglês *Auto-Guided Vehicle,* veículo auto-guiado, e é uma solução para o transporte automatizado de manufatura dentro de uma instalação.

Tempelmeier e Kuhn (1993) apresentam os tipos de flexibilidade que podem ser obtidos em um FMS. Segue os aspectos de flexibilidade relacionados aos elementos de um FMS:

- *Flexibilidade de Máquina:* descreve a facilidade com que uma máquina pode variar de uma operação a outra. Por exemplo, a troca, na máquina, de uma ferramenta por outra localizada em um *magazine de ferramentas* local. Caberia neste contexto, a discussão de quão rápido é o processo de configuração de uma máquina durante essa mudança com as instruções para a realização da nova tarefa (o chamado tempo de *setup*).
- *Flexibilidade ao lidar com materiais:* é a habilidade de um FMS na manipulação e movimentação de peças e na localização das mesmas. É influenciado pelo projeto técnico e *layout* dos caminhos de transporte.
- *Flexibilidade de Operação:* é a possibilidade de haver tipos de peças capazes de serem processadas por diferentes tecnologias e distintas seqüências de operações (planos de processo). Quanto maior a flexibilidade de operações melhor pode ser a distribuição de recursos entre as máquinas, aumentando o potencial de produção em um FMS em termos de volume de produção.

Toda essa flexibilidade traz uma série de benefícios como discute Groover (1987), implicando inclusive a flexibilidade da programação da produção. Porém, nestas condições, encontrar uma programação que permita atingir certos objetivos, torna-se uma tarefa bastante complexa.

2.4 Planejamento e Controle da Produção

Programação é parte do planejamento e controle da produção (PCP), que é a atividade, em um sistema de manufatura, na qual se define metas e estratégias, formula planos para atingi-las, administra recursos humanos e físicos com base nestes planos, direciona a ação dos recursos humanos sobre os físicos e acompanha esta ação, permitindo a correção de prováveis desvios (Tubino, 2000).

As atividades de PCP envolvem decisões de longo, médio e curto prazo. No longo prazo, os gerentes de produção utilizam previsões de demanda, disponibilidade de recursos descritos de forma geral e seus objetivos são estabelecidos em grande parte em termos financeiros. No médio prazo, um maior grau de detalhamento está presente, como por exemplo, os produtos demandados são classificados e quantificados e planos de contingências são definidos para permitirem pequenos desvios nos planos. No planejamento e controle de curto prazo considera-se quando, onde e em que seqüência os produtos deverão ser produzidos. Nesse estágio, muitos dos recursos terão sido definidos e as intervenções estão na tentativa de equilibrar a quantidade, a rapidez, a confiabilidade, a flexibilidade e os custos das operações *ad hoc* (Slack *et al.*, 1999). Um esquema hierárquico dessas atividades é mostrado na figura 2.5. Essas atividades são descritas a seguir.

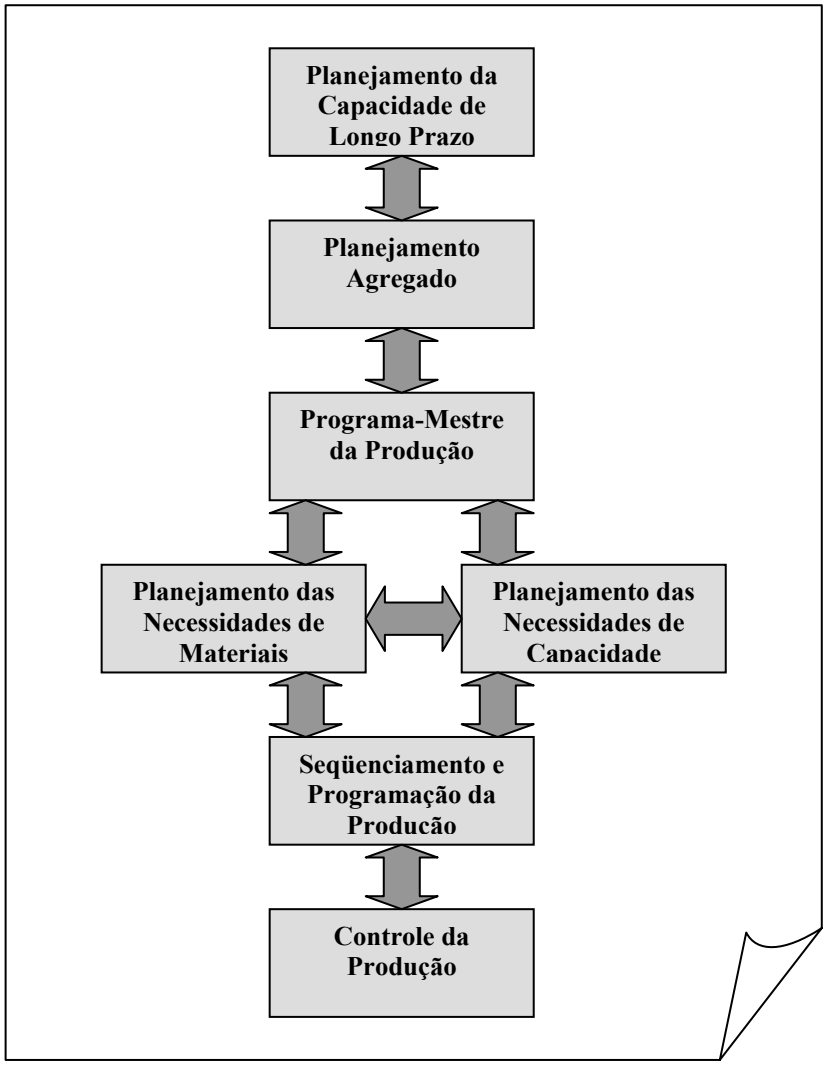

FIGURA 2.5 Atividades de PCP (Carvalho, 2003).

É chamada de *planejamento da capacidade de longo prazo*, a atividade em que são tomadas as decisões de longo prazo, tais como a determinação do tamanho, localização, *layout* e capacidade das instalações produtivas, assim como, também, os planos para fornecedores de grande porte, se aplicável, e planos de processamento, sendo que neste último estão envolvidos aspectos sobre a tecnologia de produção a ser usada, englobando formas de sistemas de automação e tipos de processo (Gaither & Frazier, 2001).

A partir do planejamento realizado de longo prazo, faz-se o *planejamento agregado*, ou seja, um plano, definido em um horizonte de longo a médio prazo, em que se estabelece: níveis de produção, dimensões da força de trabalho e níveis de estoque. Neste processo é feita a conciliação das restrições de capacidade com as previsões de demanda.

O chamado *programa-mestre da produção* (PMP) é gerado a partir do planejamento agregado de produção, desagregando-o em produtos acabados. O PMP guiará as ações do sistema de manufatura no curto prazo, estabelecendo quando e em que quantidade cada produto deverá ser produzido neste horizonte de planejamento.

Para se fazer um estudo de viabilidade de um PMP, faz-se um *planejamento das necessidades de materiais* e um *planejamento das necessidades de Capacidade*. No primeiro caso, é feito um levantamento da quantidade necessária de material de determinado tipo necessária e do tempo necessário para que os mesmos estejam disponíveis para o uso. No planejamento das necessidades de capacidade, a carga de trabalho (volume e tempo) e o tempo por centro de processamento é computado para ver se o PMP poderá ser cumprido. Se, por meio dessas duas atividades, for concluído que o PMP não pode ser executado, um novo PMP é feito baseado nas informações obtidas. Todo o ciclo se repete até se ter um PMP viável.

As etapas seguintes, de curto prazo, vêm a ser o *seqüênciamento* e a *programação da produção*. No *seqüenciamento* é definida a ordem em que lotes de determinados tipos de produtos iniciam seu processo de fabricação no chão de fábrica. Em geral, uma política de prioridades é estabelecida de forma que a produção satisfaça certas condições, tais como cumprir uma data de entrega e minimizar o tempo total de produção.

Na *programação da produção*, cada etapa de fabricação de um dado tipo de produto é distribuída ao longo do tempo entre as estações de processamento do sistema de manufatura. Tem-se assim, um cronograma definindo o momento de execução de cada uma destas operações no chão de fábrica.

A etapa de *controle* envolve a arquitetura da distribuição de informações entre as máquinas do sistema e, também, a lógica de despacho de eventos no sistema como um todo, para que as operações sejam corretamente executadas segundo o planejado.

2.5 Programação da Produção

Como definido anteriormente, a *programação da produção* é a atividade de PCP de curto prazo, restrita ao nível operacional do sistema produtivo, no caso, que concerne às operações em estações de trabalho em sistemas automatizados de manufatura.

A programação define qual operação deve ser feita em que máquina e em qual momento. Dependendo do tipo de sistema de manufatura, que é o caso do FMS, há um grande número de possibilidades para um conjunto de produtos a serem produzidos.

Enfocando a forma de construção da programação, se baseado no término ou no início das operações, pode-se ter:

- **programação para frente**: todas as operações são iniciadas o mais brevemente possível; assim, tão logo quanto houver a disponibilidade de recursos (estações de processamento e materiais), aloca-se estes para a operação compatível no momento;

- **programação para trás:** a alocação das operações em máquinas ao longo do tempo é feita baseada no prazo limite para o término das operações, visando o cumprimento da data de entrega, de forma que cada operação é executada o mais tarde possível, mas cumprindo o prazo previsto; evita-se o uso de recursos antes que sejam realmente necessários, mas, em geral, estará mais propenso a descumprir prazos na ocorrência de algum imprevisto.

Uma forma de representar graficamente a programação da produção é através do gráfico de Gantt[2]. Na figura 2.6, há dois gráficos de Gantt representando a programação para

[2] Gráfico de Gantt - diagrama de barras horizontais, que representam a duração de tarefas em relação à progressão do tempo. Sua invenção é atribuída ao engenheiro Henry Laurence Gantt, por volta de 1910.

trás e a para frente de um mesmo Plano Mestre de Produção, estando ambas com as operações associadas aos mesmos recursos e com o mesmo tempo total de produção. Nos gráficos, *t* é o tempo corrente. Cada barra representa uma operação associada a uma etapa de fabricação de um tipo de produto. As cores das barras definem os tipos de produto. O término de uma etapa de um tipo de produto é requisito para o início da etapa subseqüente. Em (b), na figura 2.6, poderia se supor que a primeira operação da máquina 1 ainda possa ser adiada, já que a máquina só voltaria a executar uma outra operação no instante *t*. Entretanto, as etapas subseqüentes a essa, respectivamente nas máquinas 3 e 4, dependem do término da primeira, que se postergadas alterariam o tempo total de processamento.

Pode haver várias possibilidades de programação para um mesmo conjunto de entradas do sistema de produção. Diferentes critérios de desempenho podem ser adotados para a escolha da programação mais adequada. Algumas medidas são normalmente usadas para estabelecer esses critérios.

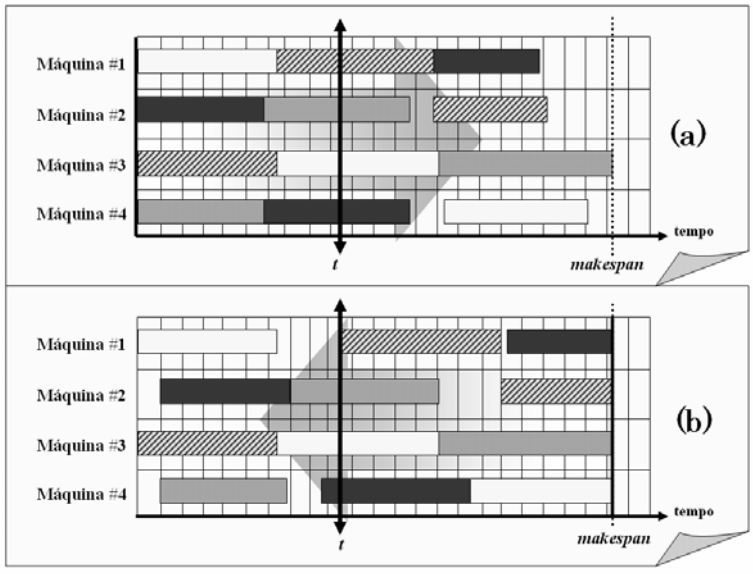

FIGURA 2.6 Gráficos: (a) *programação para frente*; (b) *programação para trás*.

Medidas de Desempenho

Na figura 2.6, tem-se um caso em que o prazo limite para término de todas as operações coincide com o tempo total de execução de todas as tarefas, pela definição de *programação para trás*. No gráfico aparece o termo *makespan*, comumente usado para se

medir o desempenho. Segue uma definição para esse e outros termos que são normalmente usados para medir desempenho e definir critérios de otimização para o problema da programação da produção de um FMS (ver capítulo 5):

- *makespan*: tempo total de processamento de um produto na fábrica, obtido no momento do término da execução da última tarefa (operação) sobre o produto no chão de fábrica;

- *Tempo de Percurso f* (*flow time*): é o tempo que um produto leva para percorrer todas as etapas de seu processamento; o tempo de percurso máximo $f_{\max} = \max\{f_1 ... f_n\}$ pode, às vezes, coincidir com o tempo de percurso total de todos os produtos, que é igual ao *makespan*; também se usa o *tempo de percurso médio*, que é a média aritmética dos tempos de percurso dos produtos envolvidas no processo,

$$f_{me} = \frac{\sum_{i=1}^{n} f_i}{n}$$

- *entidades em curso WIP* (*Work In Progress*): mede a quantidade de materiais que num certo momento estão em produção incluindo aqueles que estão em *buffers* de máquinas; é intuitivo que um grande *tempo de percurso médio* implicará um alto valor de *WIP*, assim, essas grandezas são diretamente proporcionais, com fator de proporcionalidade definido como a *taxa de produção* do sistema (*troughput* ou *output rate*);

- *grau de atraso* (*lateness*): é a diferença entre o "prazo de entrega" do produto acabado e o instante real de conclusão do mesmo; também, usa-se o *grau de atraso médio*, que é a média aritmética dos valores dos graus de atraso de todos os produtos em questão;

- *grau de atraso efetivo* [*ou positivo*] (*tardiness*): é similar ao *grau de atraso*, porém vale zero quando o tempo de conclusão é inferior ao prazo limite; na figura 2.7a e 2.7b, tem-se a representação em gráfico do comportamento desses dois parâmetros;

- *grau de adiantamento* (*earliness*): o oposto do grau de atraso, ou seja, tem, em relação a este, sinal contrário (ver figura 2.7c);

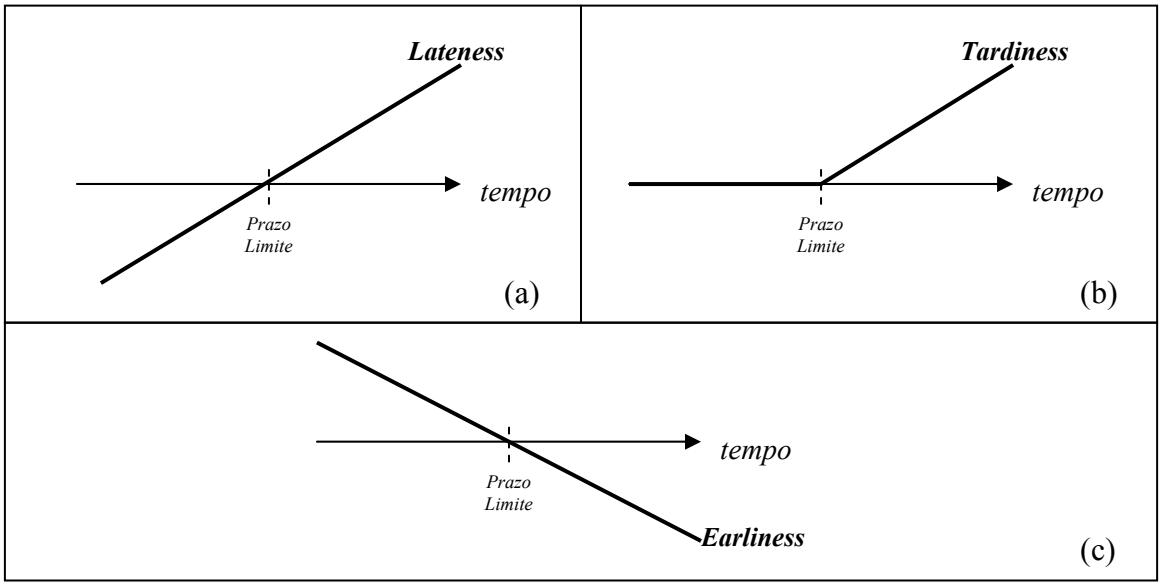

FIGURA 2.7 Representação gráfica do comportamento de (a) *lateness*, (b) *tardiness* e (c) *earliness* em relação ao tempo real de conclusão do produto.

Pode-se ter como objetivo achar uma programação com o mínimo *makespan*, ou uma com o menor *WIP*, para um menor custo de material parado. Ou, ainda, tentar reduzir o grau de atraso dos produtos, para melhor atender o cliente. Assim, de acordo com as necessidades da empresa segundo suas estratégias, um critério usando uma ou um conjunto dessas medidas pode ser adotado.

Tratando-se de FMSs, resolver a questão da otimização de uma dessas medidas é um trabalho de alta complexidade computacional.

2.6 O problema da programação de um FMS

Em um FMS, a produção se dá de forma intermitente sem padrão de fluxo definido e, mesmo para um tipo de produto, pode haver diferentes roteiros de fabricação. Cabe à programação a escolha do que melhor se ajusta às necessidades da organização em questão.

No problema geral de programação de um FMS, um número de *recursos de transformação* é considerado para a obtenção de diferentes tipos de produtos. Um conjunto de recursos é usado para uma *operação* sobre peças neste processo. Tem-se, por meio de um plano de rotas alternativas, a ordem de operações em que certas peças devem submeter-se

para, desta forma, obter um tipo de produto. Isso implica mais de uma maneira de se obter um mesmo produto, sendo que as decisões sobre esse processo podem ser tomadas para cada item considerando a produção simultânea de vários tipos de produtos por recursos compartilhados.

A programação de um FMS consiste na definição das operações ao longo do tempo que melhor satisfaça um ou mais critérios de desempenho, como, por exemplo, o mínimo *makespan*. Frente ao grande número de possibilidades, podendo envolver questões de transporte, tempo de *setup* de máquinas, possibilidades de falhas e outras coisas, tem-se um problema de grande complexidade computacional em que a obtenção de uma resposta ótima e precisa, envolvendo todos os fatores enumeráveis por um especialista da área, é no mínimo inviável ou, dependendo dos fatores considerados, talvez se configure em um problema indecidível. Certas considerações são, então, assumidas baseando-se em um modelo do sistema real, para se obter uma solução satisfatória.

Em um FMS, e em sistemas automatizados de manufatura em geral, mesmo nos que usam um processo contínuo de produção, pode-se identificar a ocorrência assíncrona de mudanças discretas ou eventos ao longo do tempo. Por esta razão, estes podem ser caracterizados como sistemas de eventos discretos. Será apresentada no próximo capítulo, uma ferramenta formal para a modelagem e análise destes tipos de sistemas.

CAPÍTULO 3 *Rede de Petri*

REDE de Petri (PN) é uma técnica matemática com descrição gráfica criada para a modelagem de sistemas baseados em eventos discretos. Foi resultado da tese de doutorado de Carl Adam Petri intitulada "Kommunikation mit Automaten" submetida à Faculdade de Matemática e Física da Universidade Técnica de Darmstadt, Alemanha, em 1962. As bases do seu desenvolvimento teórico deram-se entre os anos de 1968 e 1976 por um grupo de pesquisadores do Instituto de Tecnologia de Massachussetts (MIT), nos EUA (Murata, 1989) (Valette *et al*., 1999).

A capacidade da rede de Petri em representar sistemas em que há processos concorrentes, com necessidades de seqüenciamento e sincronização, e compartilhamento de recursos, possibilita um número de abordagens para lidar com a modelagem de FMSs para a Programação da Produção, como será visto em maiores detalhes no capítulo 4.

3.1 Conceitos Básicos

Parte da capacidade de representação de uma rede de Petri (PN) está na possibilidade de associação de elementos abstratos a *eventos* e *condições*. São estes comumente associados a, respectivamente, transições *t* e lugares *p*. A representação gráfica do lugar é um círculo, e a da transição é uma barra ou um retângulo. Na modelagem de um problema, diferentes interpretações podem ser dadas aos termos eventos e condições, como mostra Murata (1989). Os outros elementos que compõem uma PN são: arco e marca. Segue a representação visual de cada elemento:

▯	transição (*t*)
○	lugar (*p*)
→	arco
●	marca

PN é um tipo particular de grafo em que lugares podem estar unidos a transições por meio de arcos direcionados. Quando isso ocorre, há uma relação de ordem, sugerida pelo sentido da seta no grafo, podendo um lugar estar na *entrada* e/ou na *saída* de uma transição. Pode-se ponderar um arco com um número inteiro positivo *w* para representar um conjunto de

w arcos com a mesma associação entre elementos da PN. Na representação gráfica, costuma-se indicar *w* associado ao respectivo arco e, quando omitido, assume-se que $w = 1$.

Uma marcação (ou estado) *M* de uma PN é uma função que associa, a todo lugar *p* da rede, um número inteiro não negativo $M(p)$, que corresponde à quantidade de marcas (ou fichas) contidas em *p*. Uma marca é visualmente representada por um ponto.

Esses elementos são a base para se obter uma definição formal de redes de Petri (Murata, 1989; Peterson, 1977):

Definição 3.1 Uma rede de Petri é uma quíntupla $PN = (P,T, F, W, M_0)$, em que:

$P = \{p_1, p_2 \ldots p_m\}$ é um conjunto finito de lugares;

$T = \{ t_1, t_2 \ldots t_n \}$ é um conjunto finito de transições;

$F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos;

$W: F \rightarrow \aleph$ é a função de ponderação dos arcos, sendo:

$w(p,t)$ o peso do *arco de entrada* (p,t) e

$w(t,p)$ o peso do *arco de saída* (t, p);

$M_0: P \rightarrow \aleph$ é a marcação inicial da PN;

com $(P \cup T \neq \varnothing)$ e $(P \cap T = \varnothing)$,

Uma estrutura de rede de Petri é a quádrupla $N = (P,T,F,W)$, sem especificação da marcação inicial. Uma rede de Petri pode ser denotada por $PN = (N, M_0)$.

Uma simples regra compõe a base da teoria de PN: a regra de disparo de uma transição. De acordo com esta, uma transição *t* está habilitada (apta ao disparo) se todo lugar de entrada p_i tiver ao menos o número de marcas $w(p_i,t)$. Uma transição habilitada pode ou não ser disparada. No disparo da transição *t*, de cada lugar de entrada p_i, remove-se $w(p_i,t)$ marcas e, a cada lugar de saída p_j, adiciona-se $w(t, p_j)$ marcas (Murata, 1989). Esse processo é esquematizado na figura 3.1 e exemplificado na figura 3.2.

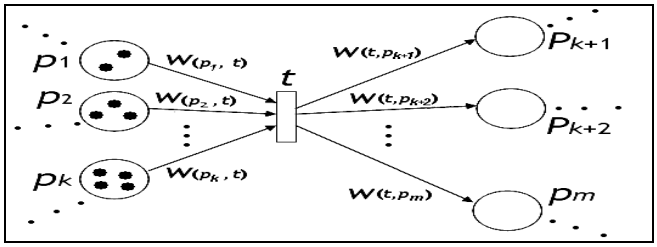

FIGURA 3.1 Esquema genérico de um trecho de uma PN para exemplificar a Regra de disparo de transição.

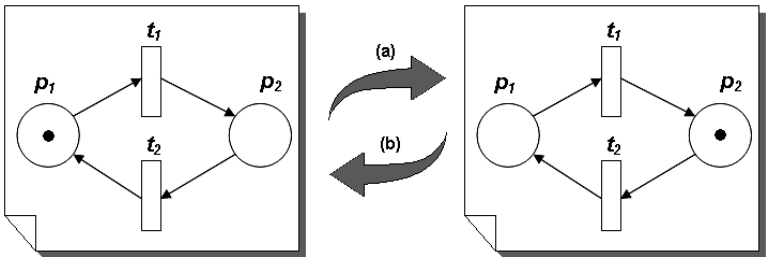

FIGURA 3.2 Mudança de estados de uma rede de Petri: (a) pelo disparo da transição t_1; (b) pelo disparo da transição t_2.

A rede de Petri, que pode ser observada na figura 3.2, é formalizada como sendo $PN_1 = (P,T, F, W, M_0)$, cujos elementos são:

$P = \{ p_1, p_2 \}$;

$T = \{ t_1, t_2 \}$;

$F = \{ (p_1, t_1), (t_1, p_2), (p_2, t_1), (t_2, p_1) \}$;

$W: F \rightarrow \aleph$, tal que $\forall f \in F$, tem-se $w(f) = 1$;

$M_0 = \{1,0\}$.

O disparo da transição t_1 de PN_1 faz com que esta mude de estado, passando da marcação inicial $M_0 = \{1,0\}$ para $M_1 = \{0, 1\}$. O disparo de t_1 gera uma nova condição que habilita t_2, cujo disparo gera a marcação $M_3 = \{1,0\}$. M_3, por sua vez, é igual a M_0, condição em que já se sabe que t_1 está habilitada.

Eventualmente, em outras estruturas PN com as respectivas marcações iniciais, pode se chegar a marcações com mais de uma transição habilitada, sendo arbitrária a escolha da ordem de disparo. Por outro lado, pode-se alcançar uma marcação na qual todas as transições encontram-se desabilitadas – o chamado *dead-end*.

Tem-se, também, o chamado *conflito* sempre que, na existência de duas transições habilitadas, o disparo de uma desabilita a outra. Neste caso, apenas uma dessas transições é disparada, escolhida arbitrariamente.

A figura 3.3 mostra um exemplo de PN que pode ser interpretada como a modelagem do uso compartilhado de um recurso por dois processos representados respectivamente por p_3 e p_5, sendo p_4 a representação da disponibilidade do recurso para o uso imediato. Essa configuração permite duas possíveis seqüências de disparo: $\sigma = t_1t_2t_3t_4$ ou $\sigma = t_3t_4t_1t_2$.

Um problema modelado em uma PN pode ser simulado através de uma seqüência finita de disparos de transições, o que permite observar o comportamento do sistema ao longo

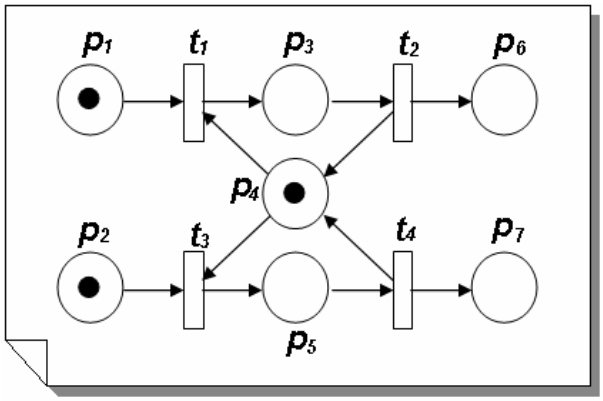

FIGURA 3.3 Uma modelagem em PN de compartilhamento de recurso.

das mudanças de marcação.

Diferentes métodos formais permitem analisar o modelo e determinar propriedades do sistema modelado, como será abordado nas próximas seções.

3.2 Propriedades

A aplicação de PN não se restringe meramente a fins descritivos, pois permite o uso de métodos para inferir no modelo propriedades que ajudam a identificar características desejáveis e indesejáveis no sistema modelado. Essas propriedades são discutidas a seguir com base nas descrições de Peterson (1977), Murata (1989) e Valette *et al*. (1999).

3.2.1 Propriedades Dependentes de Marcação

Algumas características de uma PN estão ligadas à marcação inicial M_0. Estas são chamadas de propriedades dependentes de marcação, ou comportamentais, e levam em consideração o seu dinamismo, confirmando a consistência da rede para uma marcação inicial.

3.2.1.1 Alcançabilidade

Na modelagem de um sistema em PN, pode-se lidar com as possíveis marcações do sistema, que descrevem a situação global do mesmo em um dado instante sob certas condições. Naturalmente, poderá haver situações que são desejáveis, já outras que devem ser evitadas. Isso nos remete a uma importante questão na análise de rede de Petri – é possível uma marcação ser obtida partindo-se de certas condições iniciais? Esse é o chamado *problema de alcançablilidade* (Peterson, 1977).

Uma marcação M é dita *alcançável* a partir de um estado inicial M_0 se existir uma seqüência de disparos $\sigma = t_1 t_2 ... t_N$ que transforma M_0 em M. O conjunto de todos as marcações atingíveis a partir de M_0 é denotado por $R(N, M_0)$ ou simplesmente $R(M_0)$. E o conjunto de todas as possíveis seqüências de disparo em uma rede (N, M_0) é denotado por $L(N, M_0)$ ou $L(M_0)$.

3.2.1.2 Limitação e Conservação de Marcas

Uma importante propriedade da rede de Petri diz respeito ao número máximo de marcas $M(p)$ que qualquer lugar p da rede pode conter em toda marcação M atingível a partir de um estado inicial M_0. Formalmente, define-se (Valette *et al*. 1999):

Definição 3.2 Uma rede de Petri $PN = (N,M_0)$ é dita ser k-limitada se, e somente se:

$$M(p) \leq k, \ \forall M \in R(N, M_0) \text{ e } \forall p \in P$$

Uma PN 1-limitada é chamada *segura* e, segundo Peterson (1977), corresponde a definição original de redes de Petri.

Se, na modelagem em PN, as marcas forem usadas para representar recursos, estas não podem ser criadas ou destruídas no decorrer das transições para uma modelagem consistente, em outras palavras, as marcas se conservam. Uma rede de Petri $PN = (N,M_0)$ é dita *conservativa* se o total de marcas, ou a soma das quantias ponderadas das marcas de todos os lugares p_i, da rede para toda marcação $M \in R(N,M_0)$ for constante (Peterson, 1977).

3.2.1.3 Vivacidade

A modelagem de sistemas discretos, muitas vezes, pode permitir situações de *deadlock*, que são marcações do sistema em que um conjunto de processos se encontra em *espera circular*, isto é, cada processo espera por recursos alocados por outros processos. Geralmente, este tipo de ocorrência é indesejável no sistema (Gang & Wu, 2004).

O conceito de *vivacidade* em PN está intimamente ligado à ausência de *deadlocks* nas marcações alcançáveis da rede partindo de uma marcação inicial. Seguem as definições envolvendo este conceito (Valette *et al*. 1999):

Definição 3.3 Uma transição t de rede de Petri $PN = (N,M_0)$ é dita *viva* se, e somente se, $\forall\ M \in R(N,M_0)$, $\exists\ \sigma \in L(N, M_0)$ que leva M a uma marcação M'para o qual t está habilitada.

Definição 3.4 Uma rede de Petri $PN = (N,M_0)$ é *viva* se todas as transições t de PN forem vivas.

Embora vivacidade seja uma propriedade ideal para muitos sistemas, verificá-la em alguns modelos pode ser trabalhoso e impraticável. Extensões do seu conceito podem definir diferentes níveis de vivacidade. Uma transição t de (N,M_0) é dita (Murata, 1989):

0) "*dead"* (ou morta), se não existe $\sigma \in L(N, M_0)$ em que t está habilitada;

1) *L1-viva* (potencialmente disparável), se $\exists\ \sigma \in L(N, M_0)$ em que t está habilitada;

2) *L2-viva*, se $\exists\ \sigma \in L(N, M_0)$ em que t pode ser disparada pelo menos k vezes;

3) *L3-viva*, se $\exists\ \sigma \in L(N, M_0)$ em que t aparece infinitamente;

4) *L4-viva* (ou viva) se t é L1-viva para todo $M \in R(N, M_0)$.

3.2.1.4 Reversibilidade

Uma rede de Petri $PN = (N,M_0)$ é dita reversível se é possível alcançar a marcação inicial a partir de qualquer $M \in R(N,M_0)$.

3.2.1.5 Cobertura

Uma marcação M em uma rede (N,M_0) pode ser coberta se existe M' tal que $M'(p) \geq M(p)$ para todo lugar p da rede.

3.2.1.6 Persistência

Uma PN é dita persistente se for livre de conflitos, ou seja, não existe uma marcação $M \in R(N,M_0)$ na qual, na existência de mais de uma transição habilitada, o disparo de uma desabilita a outra. Uma vez habilitada, uma transição só poderá ficar desabilitada se for disparada.

3.2.2 Propriedades Estruturais

Algumas propriedades levam em conta a estrutura topológica da PN e independem de uma marcação inicial M_0. Seguem algumas definições segundo Murata (1989):

Uma rede de Petri N é chamada de:

- *completamente controlável* se qualquer marcação for alcançável por qualquer outra marcação;
- *estruturalmente conservativa* se for conservativa para qualquer M_0;
- *parcialmente conservativa* se for conservativa para alguns M_0;
- N é *repetitiva* se existir uma marcação inicial qualquer M_0 e houver uma seqüência de disparos $\sigma \in R(N, M_0)$ tal que toda transição ocorre infinitamente em σ;
- *estruturalmente viva* se for viva para qualquer M_0;
- N é *consistente* se existir uma marcação inicial qualquer M_0 e houver uma seqüência de disparos $\sigma \in R(N, M_0)$ que retorna a M_0 – *seqüência de disparos cíclica* - tal que toda transição ocorre ao menos uma vez em σ;
- N é *parcialmente consistente* se existir qualquer M_0 e $\sigma \in R(N, M_0)$ que retorna a M_0 tal que alguma transição ocorre ao menos uma vez em σ;
- *estruturalmente limitada* se for limitada para qualquer M_0;

3.3 Métodos de Análise

Métodos formais permitem analisar uma rede de Petri e identificar propriedades presentes no modelo.

Métodos mostrados a seguir são descritos segundo Murata (1989) e Valette *et al*. (1999). São eles a árvore de cobertura (ou alcançabilidade), uma abordagem algébrica, envolvendo equações de estado e análise estrutural, e técnicas gráficas de simplificação da estrutura da rede de Petri.

3.3.1 Árvore de Cobertura x Árvore de Alcançabilidade

Em uma rede de Petri $PN=(N,M_0)$, pode-se ter a relação de precedência de toda a marcação $M \in R(N,M_0)$ numa representação em árvore - a *árvore de alcançabilidade*.

Porém, em redes não limitadas em número de marcas, esta árvore pode ser impossível de ser construída por ter ramos representando seqüências infinitas de disparo de transições. Para manter a árvore finita é introduzido o símbolo ω, que pode ser entendido como infinito e cujas características são: $\omega > n$, $\omega \pm n = \omega$ e $\omega \geq \omega$, para todo qualquer inteiro n. Isto permite construir a chamada *árvore de cobertura* segundo o algoritmo 3.1 (Murata, 1989).

Algoritmo 3.1

1) Identifica-se a marcação M_0 como "nova".
2) Enquanto "novas" marcações existirem, faça:
    2.1) Seleciona-se uma nova marcação;
    2.2) Se M for idêntica a uma marcação no caminho da raiz a M, então M é identificada como "velha" e vai para 2.1.
    2.3) Se nenhuma transição estiver habilitada em M, indentificá-la como "dead-end" e vá para 2.1.
    2.4) Enquanto existir transições habilitadas em M, faça para cada transição t:
        2.4.1) Obter marcação M' do resultado de disparo de t;
        2.4.2) No caminho da raiz até M, se existir M'' que $M'(p) \geq M''(p)$ para cada p e $M' \neq M''$ então substitua $M'(p)$ por ω em todo p em que $M'(p) > M''(p)$;
        2.4.3) Acrescentar M' como um nó, desenhar um arco direcionado, etiquetado com t, ligando M a M', e identificar M' como "nova".

A partir da árvore de cobertura ***T*** é possível obter o grafo de cobertura ***G*** = (***V***, ***E***) (ou de alcançabilidade no caso das redes limitadas), em que ***V*** são os vértices formados pelas marcações distintas presentes em ***T***, e ***E*** é o conjunto dos arcos direcionados, "etiquetados" com nomes de transições, indicando a relação de precedência entre os nós em ***T***.

Segundo Murata, algumas propriedades de uma $PN=(N,M_0)$ podem ser estudadas a partir da árvore de cobertura ***T***, como segue:

- *PN* é *limitada*, e portando o conjunto $R(M_0)$ é finito, se, e somente se, ω não aparece em nós ao longo de ***T***. Neste caso a ***T*** também é a *árvore de alcançabilidade* da *PN*.
- *PN* é *segura* se apenas zeros e uns aparecem nos nós de ***T***.
- *t* é uma *transição morta* se, e somente se, não aparece nos arcos etiquetados de ***T***.
- *M* é alcançável a partir de M_0 se houver algum *M*' em um nó de ***T*** em que $M'(p) \geq M(p)$ para todo *p*.

3.3.2 Matriz de Incidência e Equação de Estado

A análise da rede de Petri $PN = (P,T,F,W, M_0)$ em uma abordagem algébrica pode ser obtida a partir da representação de sua dinâmica em equações matriciais.

Para isso, a marcação *M* é entendida como uma matriz coluna $n\times1$, com $n = |P|$, cujos elementos são inteiros não negativos representando a quantidade de marcas em cada lugar $p_i \in P$.

Outro aspecto está na definição de *matriz de incidência* (Murata, 1989) (Valette *et al.*, 1999):

Definição 3.5 Sejam *I* e *O* funções $(P\times T) \cup (T\times P) \rightarrow \aleph$ tais que

$$I(p,t) = \begin{cases} 0, & \text{se } p \textbf{ não} \text{ for } \textit{lugar de entrada} \text{ de } t \\ w(p,t), & \text{caso contrário} \end{cases}$$

$$O(p,t) = \begin{cases} 0, & \text{se } p \textbf{ não} \text{ for } \textit{lugar de saída} \text{ de } t \\ w(t,p), & \text{caso contrário} \end{cases}$$

A *Matriz de Incidência* $C_{m\times n}$, com $n = |P|$ e $m = |T|$, é dada por:

$$c_{ij} = O(p_j,t_i) - I(p_j,t_i) \tag{3.1}$$

Desta forma, obtém-se a equação da mudança de marcação como resultado do disparo da transição t_i:

$$M_k = M_{k-1} + C^T u_k \text{, } k = 1, 2, \ldots \tag{3.2}$$

Sendo u_k uma matriz coluna $n \times 1$ em que o k-ésimo elemento é igual a 1 e todos os demais nulos, este termo representa o disparo da transição k. Desta forma, é possível reescrever a equação 3.1:

$$M_d = M_0 + C^T \sum_{k=1}^{d} u_k \qquad \text{ou} \qquad \Delta M = C^T \sum_{k=1}^{d} u_k \tag{3.3}$$

A partir desta equação é possível obter importantes propriedades, como, por exemplo, uma condição necessária para a alcançabilidade. Para apresentar esta propriedade, divide-se a matriz de incidência $C_{m \times n}$ de posto r da forma:

$$C = \begin{matrix} \overset{m-r}{\Leftrightarrow} & \overset{r}{\Leftrightarrow} \\ \end{matrix} \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{matrix} \Updownarrow r \\ \Updownarrow n-r \end{matrix} \tag{3.4}$$

Sendo C_{12} uma matriz quadrada de ordem r. Seja, ainda, I_μ a matriz identidade de ordem $\mu = m - r$. Desta forma, define-se a *matriz do circuito fundamental*:

$$B_f = [I_\mu : C_{11}^T (C_{12}^T)^{-1}] \tag{3.5}$$

Como demonstra Murata (1989), para que um estado M_d seja alcançável a partir de M_0 é necessário, mas não suficiente, que $B_f \Delta M = 0$, sendo $\Delta M = M_d - M_0$.

3.3.3 Análise Estrutural

Seja uma rede de Petri PN = (N, M_0) com matriz de incidência C. A solução da equação $C^T x = 0$, em x composto de números inteiros é chamada de t-invariante.

Além disso, há o s-invariante que é a solução da equação $Cy = 0$, em y e representa um conjunto não ordenado de disparos de transição.

Esses dois elementos – s-invariante e t-invariante – permitem verificar um número de propriedades estruturais de uma dada rede de Petri.

Nas tabelas a seguir, estão representados *t*-invariante e *s*-invariante, respectivamente, por *x* e *y*. O *i*-ésimo elemento do vetor *v* será representado por *v*(*i*); $v > w$ representa que $v(i) > w(i)$ para cada *i*; $v \geq w$ representa que $v(i) \geq w(i)$ para cada *i*, e $v \underset{\neq}{>} w$ significa que $v \geq w$ e $v(i) \neq w(i)$ para pelo menos um *i*.

Apresenta-se na Tabela 3.1 quais propriedades estruturais estarão presentes na PN mediante a constatação de certas condições envolvendo os invariantes abordados. Na Tabela 3.2 também são observadas certas características que podem ser verificadas sempre que determinada condição é satisfeita.

TABELA 3.1 Condições para algumas propriedades estruturais (Murata, 1989).

Verificam-se as Propriedades	**Condições necessárias e suficientes**
Estruturalmente Limitada	$\exists y > 0,\ C \cdot y \leq 0$ (ou $\nexists x > 0, C^T x \underset{\neq}{>} 0$)
Estruturalmente Conservativo	$\exists y > 0,\ C \cdot y = 0$ (ou $\nexists x, C^T x \underset{\neq}{>} 0$)
Parcialmente Conservativo	$\exists y \underset{\neq}{>} 0,\ C \cdot y = 0$
Repetitivo	$\exists x > 0,\ C^T \cdot x \geq 0$
Parcialmente Repetitivo	$\exists x \underset{\neq}{>} 0,\ C^T \cdot x \geq 0$
Consistente	$\exists x > 0,\ C^T \cdot x = 0$ (ou $\nexists y, Cy \underset{\neq}{>} 0$)
Parcialmente Consistente	$\exists x \underset{\neq}{>} 0,\ C^T \cdot x = 0$

TABELA 3.2 Análise de outras propriedades estruturais (Murata, 1989).

Se	**Então**
N é estruturalmente limitada e estruturalmente viva	*N* é estruturalmente conservativa e consistente
$\exists y \geq 0,\ C y \underset{\neq}{<} 0$	Não existe M_0 viva para N. N é não consistente.
$\exists y \geq 0,\ C y \underset{\neq}{>} 0$	(N, M_0) não é limitada para uma M_0 viva.
$\exists x \geq 0,\ C^T \cdot x \underset{\neq}{<} 0$	Não existe M_0 viva para a estruturalmente limitada *N*. *N* não é conservativa.
$\exists y \geq 0,\ C y \underset{\neq}{>} 0$	Não é estruturalmente limitada *N*. *N* não é conservativa.

3.3.4 Técnicas de Redução para Análise

Um conjunto de técnicas gráficas permite simplificar a estrutura de uma PN, preservando as propriedades de vivacidade, segurança e limitação. É, desta forma, facilitada a análise de modelos compostos por inúmeros elementos.

Murata (1989) apresenta seis transformações básicas mostradas na Fig. 4.3: (a) fusão de lugares em série; (b) fusão de transições em série; (c) fusão de lugares em paralelo; (d) fusão de transições em paralelo; (e) eliminação de lugares com *self-loops*; (f) eliminação de transições com *self-loops*.

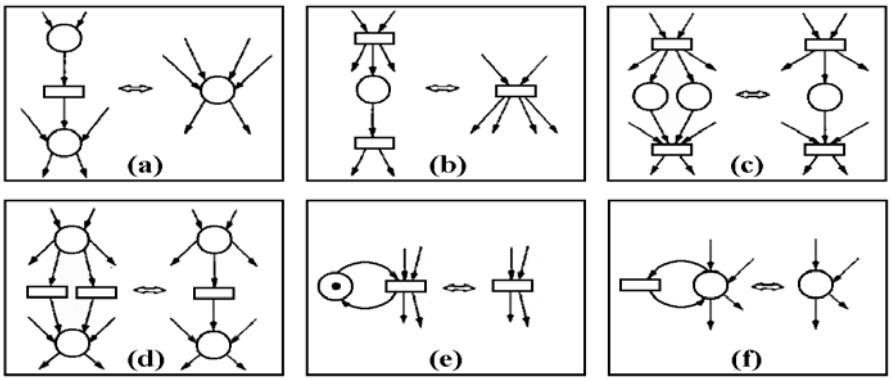

FIGURA 3.4 Algumas transformações de PN (Murata, 1989).

3.4 Redes de Petri Modificadas

"Its dificulty to model some events or conditions in systems by Petri Nets, and it has been shown that the correct modeling of other relatively resonable systems is impossible."

(Peterson, 1977)

Segundo Peterson (1977), o sucesso de qualquer modelo deve-se a dois fatores: o *poder de modelagem* e o *poder de decisão*. O primeiro refere-se à habilidade de se representar um sistema com alto grau de correção, tal que o modelo seja justamente a representação do sistema modelado. O poder de decisão está na habilidade de se analisar versões específicas do modelo e determinar propriedades do sistema modelado.

Se as transições de uma rede de Petri forem rotuladas com símbolos não necessariamente distintos, uma seqüência de disparo de transições gera uma cadeia de

símbolos. O conjunto das cadeias de símbolos geradas por todas as possíveis seqüências de disparo define uma linguagem formal chamada *Linguagem de rede Petri*. Na figura 3.5 tem-se um exemplo, extraído de Murata (1989), de um PN com rótulos nas transições, com λ representando o símbolo nulo, cujas seqüências de disparo geram uma linguagem formal $L(M_0) = \{a^n b^n c^n \mid n > 0\}$.

Foi demonstrado que *toda* linguagem de rede de Petri é uma linguagem livre de contexto (Peterson, 1981, citado por Murata, 1989). Na hierarquia de Chomsky[1], a gramática que gera esse tipo de linguagem é um caso particular de uma gramática mais ampla, capaz de ser modelada em uma máquina de Turing e que gera a chamada *linguagem recursivamente enumerada*. Isso tem implicações no poder de modelagem de PN.

Frente a certas restrições[2] e a certas dificuldades na modelagem, novas características tem sido, ao longo dos anos, acrescentadas na teoria de redes de Petri e, desta forma, novas variantes tem surgido, buscando melhorar a capacidade e a facilidade de representação de seus modelos.

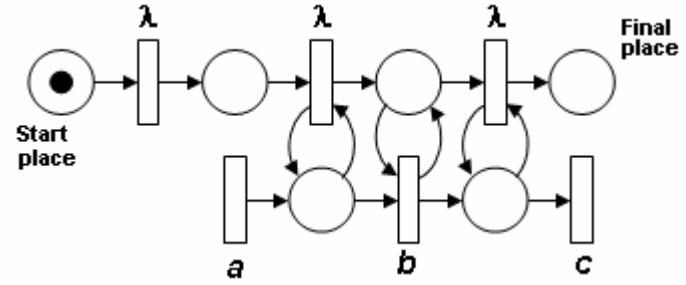

FIGURA 3.5 Modelo PN que gera a linguagem $L(M_0) = \{a^n b^n c^n \mid n > 0\}$ (Murata, 1989).

3.4.1 Variações Clássicas

A introdução de conceitos simples permite extensões de PN que podem demonstrar maior adequação a alguns tipos de problema, como descrito a seguir segundo Valette *et al.*(1999), Murata (1989) e Peterson (1977).

[1] Em 1956, Noam Chomsky criou uma hierarquia que estabelece uma relação de continência de classes de gramáticas geradoras de linguagens formais (Hopcroft, 1939).

[2] Maiores informações sobre as restrições das redes de Petri lugar/transição clássicas, consultar Agerwala e Flynn (1973); Lin (1989); Peterson (1981); Murata (1989); Busi (2002).

Uma possível abordagem sobre PN está na limitação do número de marcas que um lugar pode possuir. Em uma PN, à qual se aplica a regra de disparo previamente introduzida, há a possibilidade de um lugar conter um número ilimitado de marcas. Essas redes são chamadas de *redes de capacidade infinita*. Pode ser associada a cada lugar p uma capacidade $k(p)$, representando o número máximo de marcas que o p pode conter. Neste caso, têm-se as *redes de capacidade finita*, em que a habilitação da transição t não pode ocorrer quando existir pelo menos um lugar de saída p, cuja adição das marcas, pelo disparo de t, gere um estouro de capacidade em p, ou seja, ocasione $M(p) > k(p)$. Essa é a chamada *regra estrita de disparo de uma transição*.

Como demonstrado em Murata (1989), as redes de capacidade finita possuem o mesmo poder de modelagem da abordagem original, mas permitem uma representação mais abreviada do mesmo problema. Isso ocorre porque é possível, pela eliminação de *self-loops* seguida do método da *transformação do lugar complementar* transformar uma rede de capacidade finita em uma PN clássica. Seguem as descrições desses métodos segundo Murata (1989).

Um *self-loop* é um par (p,t) em que p é, ao mesmo tempo, lugar de entrada e de saída de uma transição t. Uma PN sem a presença de *self-loops* é chamada *pura*. O processo de eliminação de *self-loops* é mostrado na Figura 3.6.

O método da transformação do lugar complementar é o algoritmo 3.2, cuja aplicação é ilustrada na figura 3.7.

Outra extensão de PN é obtida com a introdução de um novo elemento – o chamado *arco inibidor*. Essa nova estrutura é graficamente representada por uma curva pontilhada, terminando com um pequeno círculo. Este permite, entre outras coisas, estabelecer prioridades no disparo de transições. Um arco inibidor liga um lugar p a uma transição t, desabilitando-a sempre que p contiver marcas, $M(p) \neq 0$. Isto permite o chamado *teste da condição zero* e eleva o poder de modelagem de PN ao nível de Máquina de Turing (Peterson, 1977).

Algoritmo 3.2

1) Acrescentar um lugar complementar p' para cada lugar p com $M'(p') = k(p) - M_0(p)$.

2) Entre cada transição t e algum lugar complementar p', desenhar um novo arco (t, p') ou (p',t) tal que $I(p',t)=O(p,t)$ e $O(p',t)=I(p,t)$.

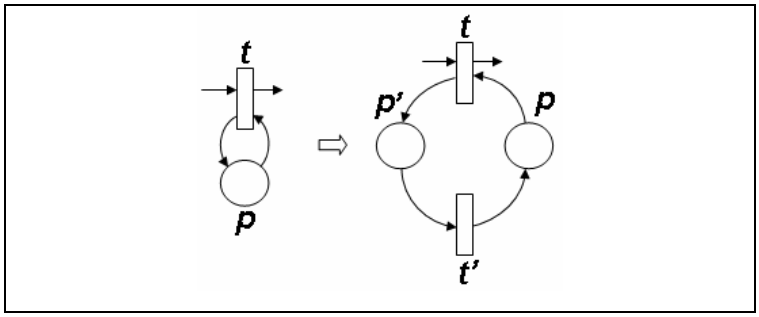

FIGURA 3.6 Transformação de *self-loop* em *loop* (Murata, 1989).

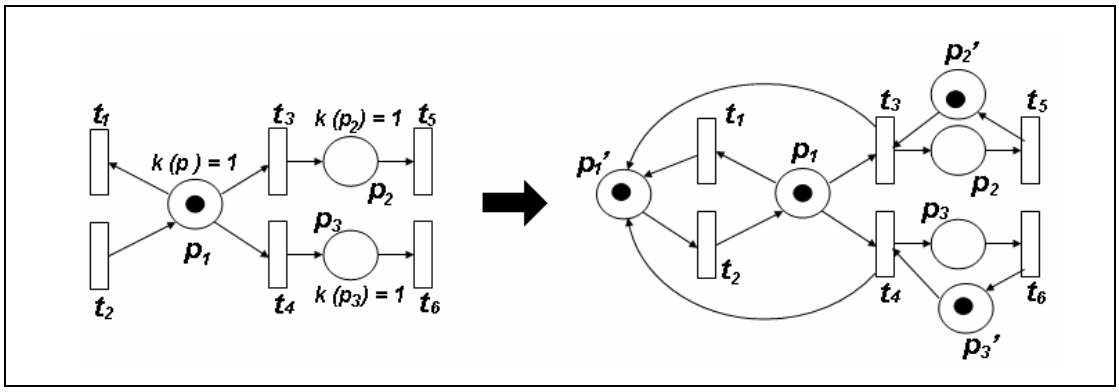

FIGURA 3.7 Aplicação da transformação do lugar complementar.

Um exemplo de uso de arcos inibidores é mostrado na figura 3.8, em que uma PN pode ser interpretada como um modelo simplificado de uma política de prioridade para processamento de dois tipos de peças por uma máquina no chão de fábrica. Outros exemplos de PN que usam essa forma de representativa de arco inibidor – tracejado e com um pequeno círculo na extremidade de chegada – são dados por Murata (1989). A presença de arcos inibidores também pode ser constatada em abordagens recentes em programação da manufatura como é abordado por Lin, Xu e Marinescu, 2001, e Lin *et al*., 2003.

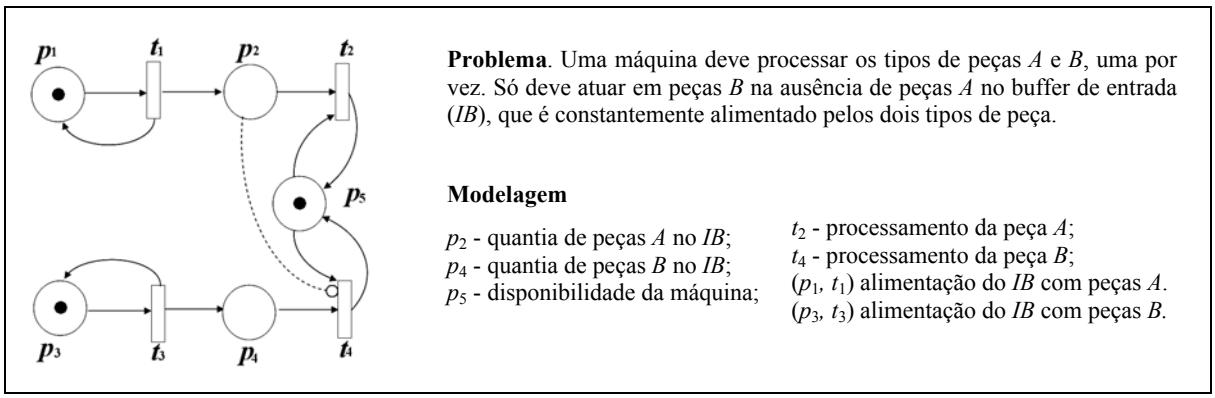

Figura 3.8 Sub-modelo em PN com política de prioridade de processamento.

Uma forma de se classificar as várias extensões de redes de Petri consiste na consideração de três grupos, dependendo de como o elemento *lugar* (*p*) é considerado. Assim sendo, têm-se as redes de nível-1 (sistemas de Condições/Eventos; sistemas de redes Elementares; sistemas 1-seguro e Máquinas Finitas de Estado), em que um lugar pode representar um valor *booleano*, ou seja, pode "conter" até uma marca não estruturada. Quando um lugar pode "conter" um número finito de marcas, podendo este inclusive representar um número inteiro não negativo, temos as redes de nível-2 ou redes lugar/transição (corresponde à abordagem da seção 3.1). Porém, os lugares podem estar associados a informações de alto nível, como se contivessem marcas que representassem estruturas de dados. São as chamadas redes de nível-3, ou redes de alto nível com tipos abstratos de dados, em que se incluem as tradicionais redes de Alto Nível (HLPN) e redes Coloridas (CPN) (Mortensen & Rölke, 2001).

3.4.2 Rede de Petri Temporizada

Extensões de PN foram criadas para lidar com tempo – neste texto referenciadas por Redes de Petri Temporizadas (TPNs). Dado que transições representam eventos, seria natural que o tempo estivesse associado às transições, representando duração de ocorrência de eventos. Porém, novas formas de se interpretar uma PN permitem, de acordo com a conveniência, atribuir uma duração de tempo a arcos ou lugares (Bowden, 2000).

Segundo Bowden (2000), há três principais abordagens na introdução do tempo em PN:

- *Durações de disparo.* Transições habilitadas podem disparar a qualquer instante. Ao iniciar o disparo, as marcas dos lugares de entrada são imediatamente removidas, mas as marcas nos lugares de saída são criadas apenas após o término do período de duração do disparo.
- *Durações de contensão.* Classificam-se as marcas em dois tipos: *disponível* e *não disponível*. Para a habilitação das transições para o disparo são consideradas apenas as marcas disponíveis. As marcas estarão disponíveis após permanecerem durante um certo tempo em um dado lugar, sendo a duração desta contensão estipulada pelas transições que as criaram. Os disparos são instantâneos.

- *Durações de habilitação*. Os disparos de transições são instantâneos. Porém, uma transição *t* não pode disparar antes de ficar habilitada ininterruptamente durante um período determinado de tempo.

3.4.3 Modularidade e Hierarquia em Rede de Petri

Um número de extensões de redes de Petri enfoca uma modelagem de forma modular e/ou hierárquica visando tornar flexível o processo de construção do modelo (PN Modulares, PN Orientadas a Objetos, redes de Petri Hierárquicas etc). Em abordagens modulares, partes do sistema são modeladas separadamente e, então, são, de certa forma, integradas para o tratamento do modelo do sistema como um todo. Visões em níveis hierárquicos também permitem auxiliar a obtenção de um modelo final em diferentes níveis de detalhamento.

Por exemplo, uma forma de se trabalhar com hierarquia em redes de Petri está na obtenção de visões em diferentes níveis de abstração pelo encapsulamento de partes da rede. Uma forma de aplicação dessa idéia são PN com possibilidade de expansão de seus elementos em uma nova sub-rede, como ilustrado na figura 3.9.

No aspecto modular, tem-se o exemplo do uso de blocos que podem ser anexados para montar a estrutura da rede de acordo com a necessidade. Esse conceito está presente, por exemplo, numa abordagem, inspirada no paradigma orientado a objetos, que usa elementos estruturais, representados por triângulos, para conexão de partes da rede, como pode ser observado na figura 3.10.

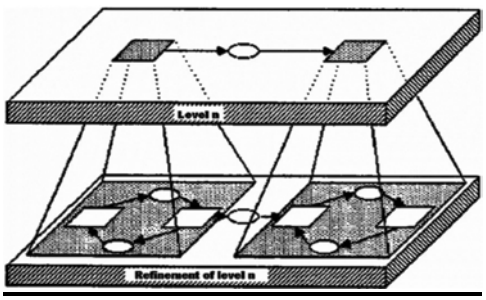

FIGURA 3.9 Rede de Petri Hierárquica (Eichenauer, 1996).

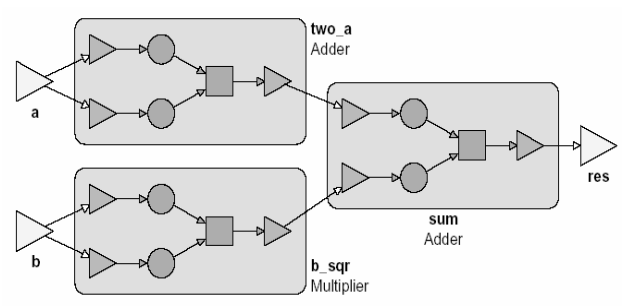

FIGURA 3.10 Rede de Petri Orientada a Objetos (Esser, 1997).

Rede de Petri Virtual

Rede de Petri Virtual consiste em abordagem de modelagem modular usando PN concebida para tratar sistemas de manufatura. Elementos de Sistemas Flexíveis de Manufatura são primeiramente modelados a parte, como por exemplo, máquinas, AGVs e/ou rotas alternativas. Então um método faz a junção entre os módulos envolvidos para obter uma rede de Petri representando o sistema como um todo (Morandin Jr., 1999).

A rede de Petri Virtual é baseada na rede de Petri Lugar-Transição, com o acréscimo de alguns elementos – os aqui chamados de *elementos virtuais*. Estes elementos são a base para a junção dos módulos.

Definição 3.6 Uma rede de Petri Virtual é formalmente definida como a quíntupla $VPN = (VP, VT, VF, VW, VM_0)$, em que:

$VP = vP \cup P$, é um conjunto virtual finito de *lugares*;

$VT = vT \cup T$, é um conjunto virtual finito de transições;

$VF = vF \cup F$, é um conjunto virtual finito de *arcos*;

$VW : VF$, é o conjunto virtual de pesos;

$vP = \{vp_{1+n}, vp_{2+n}, ..., vp_p\}$ é um conjunto finito de *lugares virtuais*;

$vT = \{vt_{1+m}, vt_{2+m}, ..., vt_q\}$ é um conjunto finito de *transições virtuais*;

$vF \subseteq (VP \times VT) \cup (VT \times VP) \cup (P \times VT) \cup (VT \times P) \cup (VP \times T) \cup (T \times VP)$ é um conjunto finito de arcos;

$VW : VF \rightarrow \{1, 2, 3, ...\}$ é a função virtual de pesos;

$VM_0 : VP \rightarrow \{0, 1, 2, 3, ...\}$ é o estado virtual inicial;

$P = \{p_1, p_2, ..., p_m\}$ é o conjunto dos *lugares* não virtuais;

$T = \{t_1, t_2, ..., t_n\}$ é o conjunto das *transições* não virtuais;

$F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos não virtuais;

$VP \cup VT \neq \emptyset$, $VP \cup T \neq \emptyset$, $P \cup VT \neq \emptyset$, $P \cup T \neq \emptyset$;

$VP \cap VT = P \cap VT = VP \cap T = P \cap T = \emptyset$.

Também é considerado que:

$\bullet p_i$ = conjunto de transições de entrada de um dado lugar p_i ;

$\bullet vp_j$ = conjunto de transições de entrada de um lugar virtual vp_j ;

$v(\bullet vp_j)$ = conjunto de transições virtuais de entrada do lugar virtual vp_j ;

$v(\bullet p_i)$ = conjunto de transições virtuais de entrada do lugar p_i ;

$p_i\bullet$ = conjunto de transições de saída do lugar p_i ;

$vp_j\bullet$ = conjunto de transições de saída de um lugar virtual vp_j ;

$v(vp_j\bullet)$ = conjunto de transições virtuais de saída do lugar virtual vp_j ;

$v(p_i\bullet)$ = conjunto de transições virtuais de saída do lugar p_i .

Esses conjuntos de vértices são mostrados na figura 3.11.

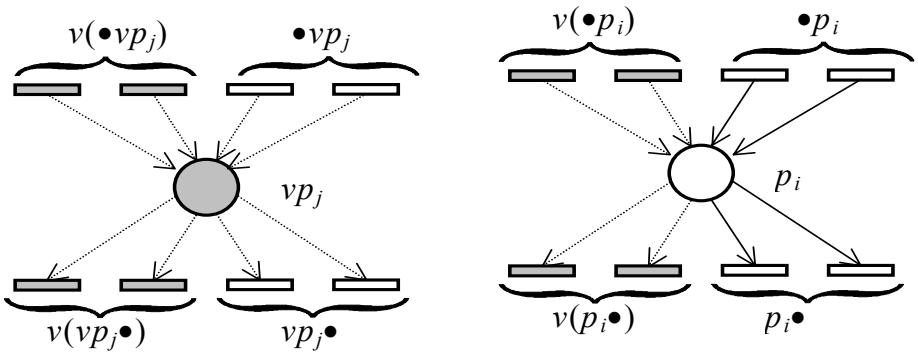

FIGURA 3.11 Vértices – lugares e transições em uma Virtual PN (Morandin & Kato, 2003).

Na figura, os elementos representados graficamente preenchidos em cinza e arcos pontilhados são elementos virtuais, em contraste com os demais elementos não virtuais presentes. As cores, embora não presentes na especificação formal do modelo, foram introduzidas para enfatizar os diferentes elementos presentes.

Da mesma forma, considera-se:

$\bullet t_i$ = conjunto dos lugares de entrada da transição t_i ;

$\bullet vt_j$ = conjunto dos lugares de entrada da transição virtual vt_j ;

$v(\bullet vt_j)$ = conjunto dos lugares virtuais de entrada da transição vt_j ;

$v(\bullet t_i)$ = conjunto dos lugares de entrada da transição t_i ;

$t_i\bullet$ = conjunto dos lugares de saída da transição t_i ;

$vt_j\bullet$ = conjunto dos lugares de saída da transição virtual vt_j ;

$v(vt_j\bullet)$ = conjunto dos lugares virtuais de saída da transição virtual vt_j ;

$v(t_i\bullet)$ = conjunto dos lugares virtuais de saída da transição t_i .

Assim, a rede de Petri Virtual pode ser usada para modelar cada elemento de um FMS, obtendo diferentes módulos.

Cada módulo é praticamente uma rede de Petri Lugar-Transição acrescida de alguns elementos virtuais (lugares, transições e arcos). Para a identificação de um modulo *a*, por exemplo, usa-se para o lugar *i*, a notação $p_{a|i}$ e, para uma transição do mesmo módulo, $t_{a|i}$ (ver figura 3.12).

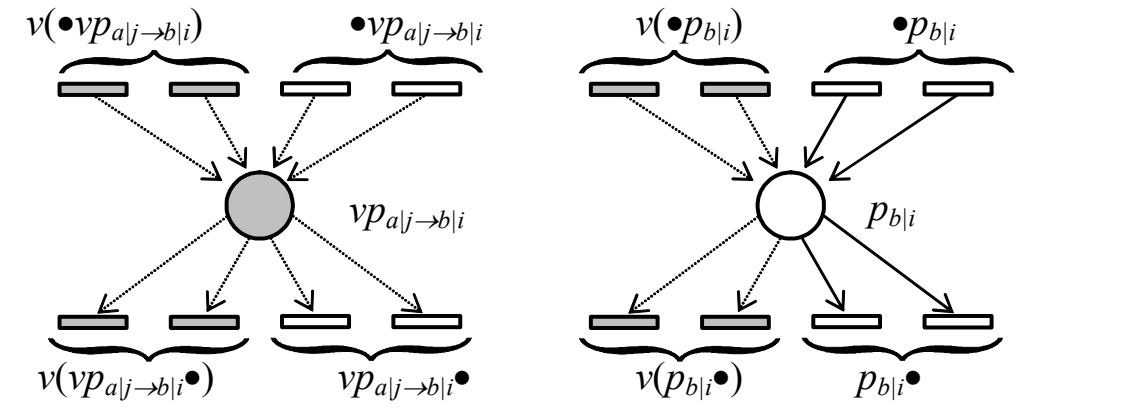

FIGURA 3.12 Módulos de rede de Petri Virtual.

A notação $vp_{a|j\to\ b|i}$ é a forma de se representar um lugar virtual vp_i que é um elemento do módulo vPN_a e tem como *alvo* p_j no módulo vPN_b .

Desta forma, o vértice virtual (lugar ou transição) é aquele que possui um *alvo* em outro módulo.

Cada módulo é construído de forma independente, originariamente representando um elemento de FMS (a motivação da criação de VPN foi a modelagem completa de um FMS).

Ligando os módulos

O processo de junção dos módulos é feito considerando-se separadamente dois a dois, ou seja, o resultado de junção dos dois primeiramente considerados é ligado ao terceiro módulo e assim por diante. Assim, une-se sempre dois módulos para a obtenção de um novo.

Uma situação é a ligação de um lugar virtual e um lugar *alvo* (não virtual). Isso pode ser definido em uma equação do tipo:

$$(vp_{aj\to b|i}) \oplus (p_{b|i}) = p_{ab|ij}$$

em que:

$vp_{a|i\to b|i}$ = lugar virtual p_i do modulo a contendo o alvo p_j no módulo b;

$p_{b|j}$ = lugar *j* do modulo *b* ;

$p_{ab|ij}$ = novo lugar após a junção.

Também, tem-se que:

$$\bullet p_{ab|ij} = \bullet vp_{a|j\to b|i} \cup \bullet p_{b|j} \cup v(\bullet p_{b|j}) \cup v(\bullet vp_{a|j\to b|i}) \ ;$$

$$p_{ab|ij}\bullet = vp_{a|j\to b|i}\bullet \cup p_{b|j}\bullet \cup v(p_{b|j}\bullet) \cup v(vp_{a|j\to b|i}\bullet) \ .$$

É possível observar algumas situações de junção na Figura 3.13.

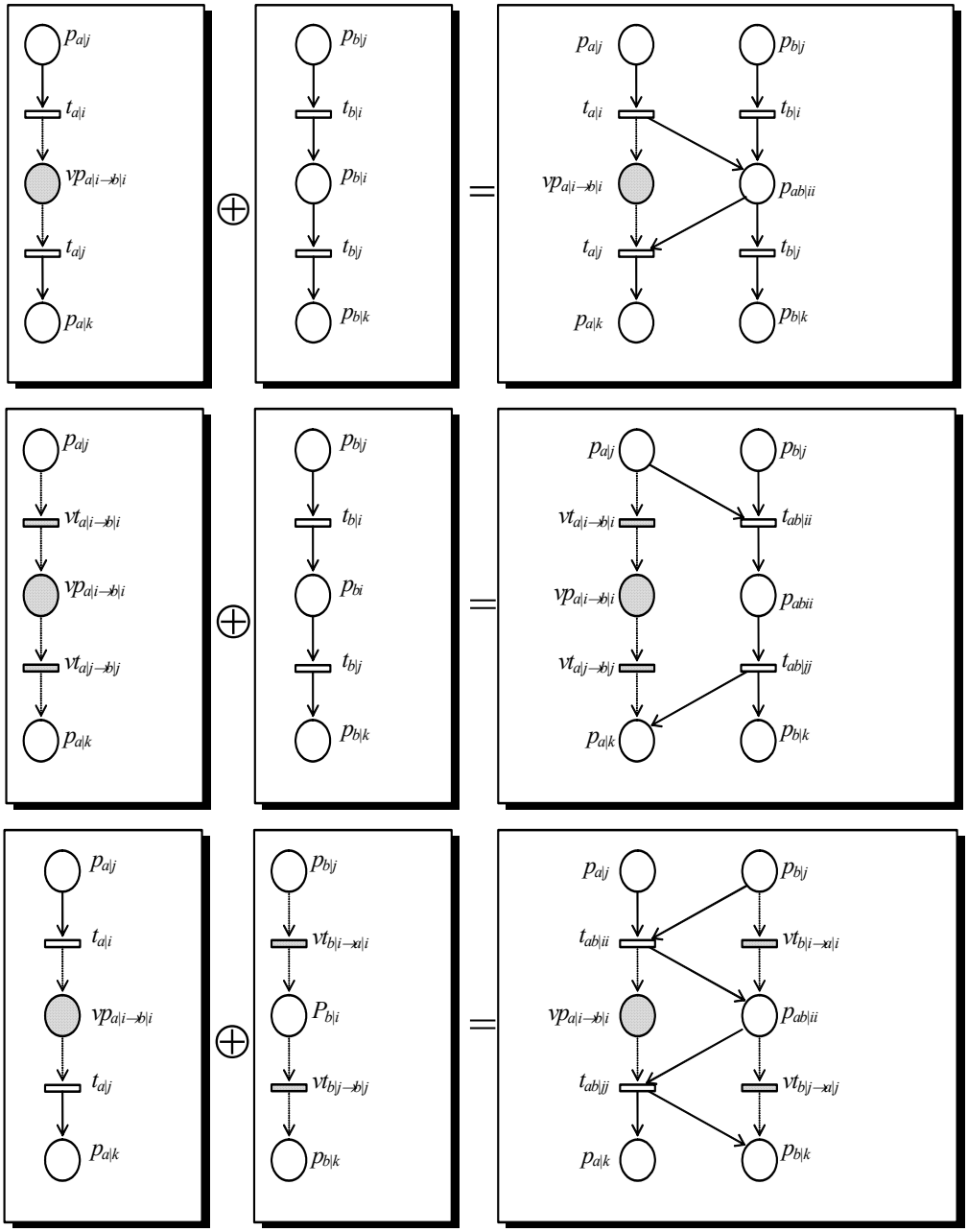

FIGURA 3.13 Exemplos de junção de módulos (Morandin & Kato, 2003).

Na figura anterior, os resultados preservam os elementos virtuais para mostrar como se deu o processo, mas na verdade eles são suprimidos para a obtenção do modelo final.

Outra possibilidade é considerar dois nós virtuais, com respectivas referências de nós alvos cruzadas entre si, na obtenção de um novo nó não virtual.

$$(vp_{a|j\rightarrow b|i}) \oplus (vp_{b|i\rightarrow a|j}) = p_{ab|ij}$$

em que:

$vp_{a|j\rightarrow b|i}$ = lugar virtual p_j do módulo a, possuindo o alvo p_i no módulo b.

$vp_{b|i\rightarrow a|j}$ = lugar virtual p_i do módulo b, possuindo o alvo p_j no módulo a.

$p_{ab|ij}$ = novo lugar obtido após a junção (de p_i).

$$\bullet p_{ab|i} = \bullet vp_{a|j\rightarrow b|i} \cup \bullet vp_{b|i\rightarrow a|j} \cup v(\bullet vp_{a|j\rightarrow b|i}) \cup (\bullet vp_{b|i\rightarrow a|j})$$

$$p_{ab|i}\bullet = vp_{a|j\rightarrow b|i}\bullet \cup vp_{b|i\rightarrow a|j}\bullet \cup v(vp_{a|j\rightarrow b|i}\bullet) \cup v(vp_{b|i\rightarrow a|j}\bullet)$$

Segue alguns exemplos deste tipo de ligação (figura 3.14) para algumas situações.

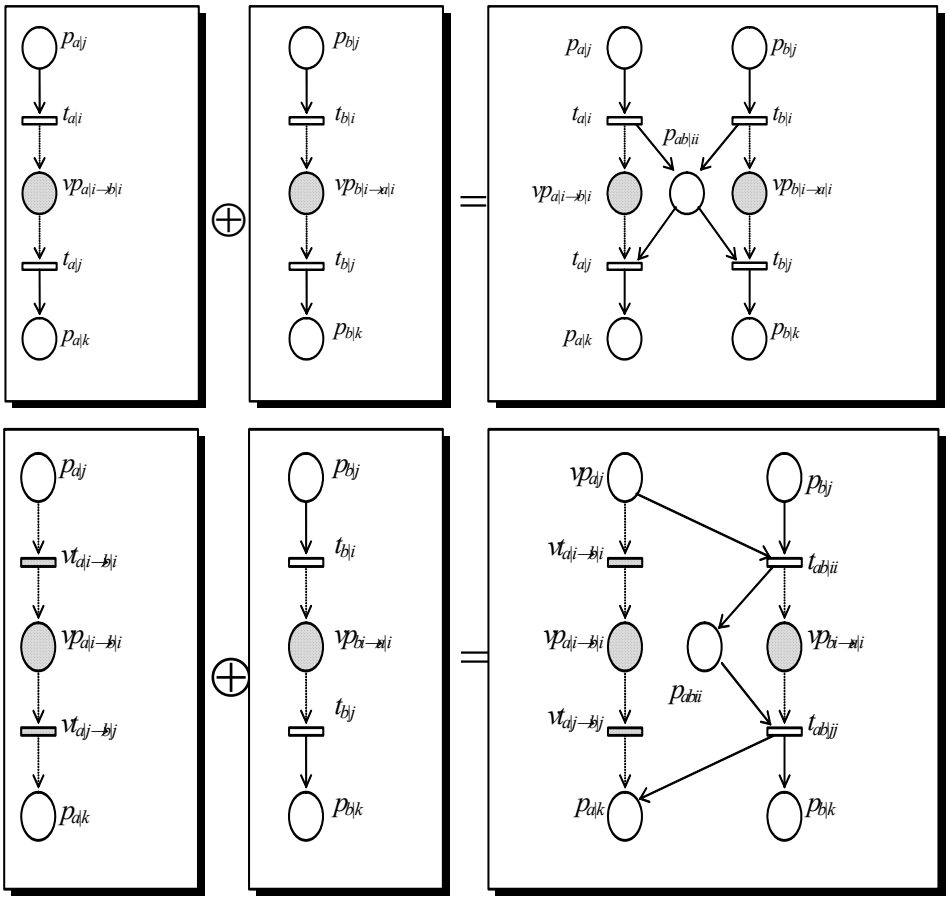

FIGURA 3.14: Exemplo de junção de módulos (2) (Morandin & Kato, 2003).

Considerando os módulos de ligação de $VPN_a \oplus VPN_b$, o modelo obtido é também uma quíntupla $VPN_{ab} = (VP_{ab}, VT_{ab}, VF_{ab}, VW_{ab}, VM_{0ab})$ em que:

$VP_{ab} = P_{va \to vb} \cup P_{vb \to va} \cup VP_{va \to vb} \cup VP_{vb \to va} \cup VP_a \cup VP_b$

$VT_{ab} = T_{va \to vb} \cup T_{vb \to va} \cup VT_{va \to vb} \cup VT_{vb \to va} \cup VT_a \cup VT_b$

$VF_{ab} = VF_{ab} \cup F_{ab}$

$VW_{ab} : VF_{ab}$

$VM_{0/ab} : VP_{ab} \to \{0, 1, 2, 3, ...\}$ é o estado inicial;

$VP_{ab} \cup VT_{ab} \neq \varnothing$ and $VP_{ab} \cap VT_{ab} = \varnothing$

Há uma outra possibilidade de junção. Se há dois nós virtuais não recíprocos, isto é, o primeiro tem como alvo o segundo, mas este, por sua vez, tem um outro nó (virtual ou não) como alvo. Neste caso a idéia aplicada é a mesma, mas desta vez resultado em um nó virtual com alvo no terceiro nó em questão.

CAPÍTULO 4 *Estratégias de Busca para Solução de Problemas*

"Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;
Then took the other"

ROBERT FROST

(do poema "The Road Not Taken")

"Personally, I think heuristic search is one of the technologies at the heart of Artificial Inteligence."

NILS NILSSON, 2004

(renomado pesquisador, autor de vários livros na área de IA)

ESTRATÉGIAS de solução baseadas na exploração de estados são usadas para tratar determinados problemas que requerem a avaliação de um grande número de possibilidades para serem resolvidos.

Em geral, tratar este tipo de problema envolve decisões na obtenção de novas configurações, novos *estados*, partindo de uma condição inicial a fim de se chegar a um *objetivo* que, no caso, pode ser qualquer dos elementos de um conjunto de estados, do modelo do mundo em questão, que satisfazem uma determinada condição que se supõe determinar a solução do problema. Tem-se, como exemplo, o problema de alcançabilidade de redes de Petri, que investiga se uma dada marcação M pode ou não ser alcançada partindo de um estado inicial M_0. Este caso, também investigado por meios analíticos (como visto no capítulo 3), pode ser tratado por exploração de estados em que o objetivo é obter M na expansão dos nós da árvore de alcançabilidade, averiguando, assim, a sua presença no universo de marcações $R(M_0)$.

Entretanto, há casos em que o objetivo na resolução do problema não é satisfazer uma condição, e sim encontrar a melhor forma de se chegar até ela. São os chamados problemas de otimização, em que interessa a *solução ótima*, ou seja, o percurso menos custoso na obtenção de uma condição, segundo algum critério numérico. Por exemplo, em rede de Petri temporizada, pode-se indagar qual a seqüência de disparos de transições, partindo de um estado M_0, leva a M no menor tempo possível.

Segue uma abordagem de formalização de problemas para uma conveniente generalização do uso das estratégias de solução que serão tratadas mais adiante.

4.1 Formalização de problemas

Um problema, no contexto abordado, pode ser entendido como uma composição de quatro elementos básicos (Russell & Norvig, 2004):

- *Estado inicial* q_0 do problema, uma condição de partida no processo de investigação da solução;

- *Ações* que são "operadores de estados" que, quando aplicados a um dado estado q_n, uma nova configuração q_{n+1} é atingida. Assim, um conjunto de ações e o estado inicial q_0 de um problema definem o *espaço de estados* $R(q_0)$, que é o conjunto dos estados alcançáveis por uma seqüência de ações a partir de q_0. Seja Υ o conjunto das ações aplicáveis sobre os possíveis estados do problema. Assim, define-se a função *sucessor*: $(R(q_0) \times \Upsilon) \rightarrow R(q_0)$ que relaciona os pares ordenados (q_n, y_k) ao elemento q_{n+1}, ou seja, q_{n+1} é sucessor de q_n pela ação y_k. O espaço de estados forma um grafo em que os vértices são os estados e os arcos são as ações. No grafo, a direção das setas dos arcos define a relação de precedência entre nós de estados, representando a função "sucessor". Um *caminho* no espaço de estados é um conjunto de estados conectados por uma seqüência de ações.

- *Teste de objetivo*, que indica se uma certa condição de parada foi atingida, podendo esta determinar o fim de um percurso na resolução de problemas de otimização ou a indicação de que a solução foi encontrada, no outro caso.

- Uma função *custo de caminho* $g(q_n)$**,** que atribui um valor numérico a cada percurso, segundo algum critério de desempenho escolhido para o problema. Pode ser entendido como a soma dos custos individuais de cada ação, de uma seqüência, executada no processo de sucessão de estados. Assim, define-se o *custo de passo* $c(q, y, q')$ como sendo o custo da ação o em q na obtenção de q'.

As ações do problema, no escopo deste texto, serão consideradas determinísticas, ou seja, supõe-se ser impossível a obtenção de mais de um estado q' a partir da execução de uma ação y sobre um dado estado q. Tal consideração simplifica o processo de obtenção do espaço de estados, ou uma parte dele.

O espaço de estados $R(q_0)$ pode, também, ser representado por uma árvore cujos vértices são ocorrências de estados e os arcos estabelecem a relação de precedência na sucessão de estados segundo uma seqüência de execuções de ações, em que os elementos sucessores e predecessores de um nó são chamados, respectivamente, de nós pais e nós filhos. Permitindo que dois *nós de estados* distintos representem um mesmo estado e restringindo, assim, o número de predecessores por nó a uma unidade, é intuitiva a obtenção de representação em árvore a partir do grafo de estados. Esta representação, neste contexto, é chamada de *árvore de busca*. O nó *raiz* (*s*), referente ao estado q_0, é o que não possui

predecessores, todos os demais possuem exatamente um nó pai. Os nós sem sucessores são chamados de terminais ou folhas. Inclui-se, também, nesta idéia o conceito de *profundidade* de um nó que está relacionado com a sua distância à raiz em número de arcos entrepostos.

A construção da árvore de busca é feita pela *expansão* de nós, que é o processo de *geração* de um conjunto de nós partindo-se de um nó específico. Os elementos desse conjunto correspondem aos estados sucessores oriundos das ações aplicáveis ao estado do nó pai.

4.2 Métricas de desempenho de resolução de problemas

Existem algumas questões que devem ser consideradas ao se tratar de estratégias de resolução de problemas, como levantadas por Russell e Norvig (2004). Primeiramente, nem todo algoritmo terá como saída uma solução. Alguns podem nem apresentar uma saída (se o curso no processo de solução induzir a uma seqüência de passos em laço de repetição infinita). Considera-se, então, a *completude* no processo de solução, que trata a questão do fato desse algoritmo encontrar ou não a solução, caso ela exista.

Outro aspecto a ser considerado está na capacidade do algoritmo de encontrar a *solução ótima*, ou seja, a que apresenta o caminho de menor custo. A relevância desta questão torna-se evidente em problemas de otimização.

Por outro lado, mesmo um procedimento com a eficácia nestes pontos abordados pode ser comprometido se apresentar baixíssimo desempenho e um alto custo computacional. Neste contexto, discute-se a *complexidade* do algoritmo envolvido. Há a complexidade no tempo, que está relacionada com o período necessário para que o algoritmo retorne uma solução, e a complexidade no espaço, reportando-se à memória requerida pelo procedimento de resolução.

A métrica usual para a complexidade é o tamanho do grafo de estados no caso do tempo e o número de nós requeridos em memória no caso do espaço. Assim sendo, a complexidade pode ser expressa em termos de três quantidades, como mostram Russel e Norvig (2004). São elas: o *fator de ramificação* (*b*) ou número máximo de sucessores por nó; a profundidade do nó alvo mais raso; o comprimento máximo (*m*) de qualquer caminho no espaço de estados.

Esses parâmetros, além de positivos, serão considerados finitos, que é conveniente dentro do escopo do texto.

O desempenho de um algoritmo de busca pode ser avaliado pelo seu custo de busca, em termos de tempo e ocupação de memória, e o custo do caminho da solução encontrada.

4.3 Processo de busca

Busca é a técnica de expansão de nós do espaço de estados de um problema, partindo de um nó raiz, construindo-se, assim, uma árvore de busca. O processo pára quando é alcançado algum nó objetivo ou quando todo o espaço de estados já estiver sido explorado sem sucesso. Pode haver outras condições de parada, como, por exemplo, o estabelecimento de um limite máximo de profundidade na árvore de busca. O exemplo clássico de algoritmo de busca pode ser resumido na seguinte seqüência de passos:

1) Iniciar uma lista com um único nó ainda a ser explorado;
2) Enquanto houver nós na lista a serem explorados, repetir:
    2.1) Retirar um nó *n* da lista e verificar se é solução;
    2.2) *Se* for solução,
        então encerrar a busca e retornar a resposta;
        caso contrário, gerar seus sucessores e inseri-los na lista.

Segue, no algoritmo 4.1, a elaboração dessa idéia, em que são usandas duas listas: *UNEXPLORED* e *EXPLORED*[1] para abrigar, respectivamente, os nós a serem explorados e os já explorados (na literatura é comum serem referidas, respectivamente, como *OPEN* e *CLOSED*). O símbolo "∪", neste contexto, representa a agregação de listas por justaposição.

Algoritmo 4.1

```
UNEXPLORED := { s };  EXPLORED := { };
while ( UNEXPLORED ≠ { } ) do
begin
    q' :=  remove_first_node_from (UNEXPLORED);
    if ( q' = <target> )
       then stop_and_return_solution ( q');
       else begin
               Q := successores_of ( q');
               UNEXPLORED := Q ∪ UNEXPLORED;
               EXPLORED := EXPLORED ∪ { q'};
            end;
end;
```

[1] Optou-se por manter os termos usados no algoritmo em inglês para uma coesão com comandos em "pseudo-pascal", visando uma melhor legibilidade.

Há casos em que a condição de parada já define de antemão um estado objetivo específico do problema. E se, além disso, ainda for possível obter uma função que forneça os predecessores de cada nó de estado, pode-se, então, fazer uma busca na direção inversa, em que o estado de partida seja a condição final do problema, objetivando-se chegar à condição inicial. Assim têm-se, de acordo com a direção, a *busca adiante* (*foreward search*) e a *busca para trás* (*backward search*).

Há também um processo em que é feita a exploração de estados simultaneamente nas duas direções sendo a condição de parada a obtenção de um nó em comum. Trata-se da *busca bidirecional*. Será abordada, mais adiante, um exemplo de uso deste conceito.

Além disso, baseado na presença ou não de informações extrínsecas ao algoritmo de busca em si, classificam-se as estratégias em: *busca heurística (*ou *busca com informação*) e *busca sem informação*.

4.4 Busca sem informação (*Blind Search*)

As estratégias de busca sem informação (ou *blind search* - "busca cega") se baseiam na obtenção da *árvore de busca*, tendo em foco os *testes de objetivo* e, no caso de problemas de otimização, a comparação de *custos de caminho*, quando possível. A busca cega apresenta variações quanto à ordem de exploração dos nós.

4.4.1 Busca em largura (*Breadth First Search*)

Na busca em largura, todos os nós de um nível de profundidade na árvore de busca são expandidos e, só então, inicia-se a expansão dos nós no nível subseqüente.

O processo é aquele previamente descrito no algoritmo 4.1. Na figura 4.2, é esquematizada a ordem de exploração de nós em uma árvore simples nesse processo por meio de setas tracejadas.

Este processo de busca é completo, pois acha uma solução sempre que ela existir. Se os custos de passo da árvore forem todos idênticos, então esta estratégia também acha a solução ótima. Tanto a complexidade no tempo como no espaço, são $\mathbf{O}(b^{d+1})$, como discutido em Russell e Norvig (2004).

4.4.2 Busca de custo uniforme (*Uniform Cost Search*)

Busca de custo uniforme estabelece a ordem de exploração começando sempre pelo nó de menor custo de caminho $g(n)$. Ele é similar à busca em largura, mas em vez de expandir primeiro os nós menos profundos, expande os menos custosos (ver algoritmo 4.2).

Sua completude e seu caráter ótimo são garantidos se existe um número positivo ε tal que o custo de caminho de cada passo seja maior ou igual a ele.

Seja Ht^* o custo da solução ótima. A complexidade no tempo e no espaço da busca de custo uniforme poderá ser, no pior caso, $\mathbf{O}(b^{[Ht^*/\varepsilon]})$, que pode ser muito maior que b^d (Russell & Norvig, 2004).

Algoritmo 4.2

```
UNEXPLORED := { s };  EXPLORED := { };
while ( UNEXPLORED ≠ { } ) do
 begin
     q' := remove_from_UNEXPLORED_the_minimum_g(n)_node;
     if ( q' = <target> )
        then stop_and_return_solution ( q');
        else begin
                Q := successores_of ( q');
                UNEXPLORED := Q ∪ UNEXPLORED;
                EXPLORED := EXPLORED ∪ { q'};
             end;
 end;
```

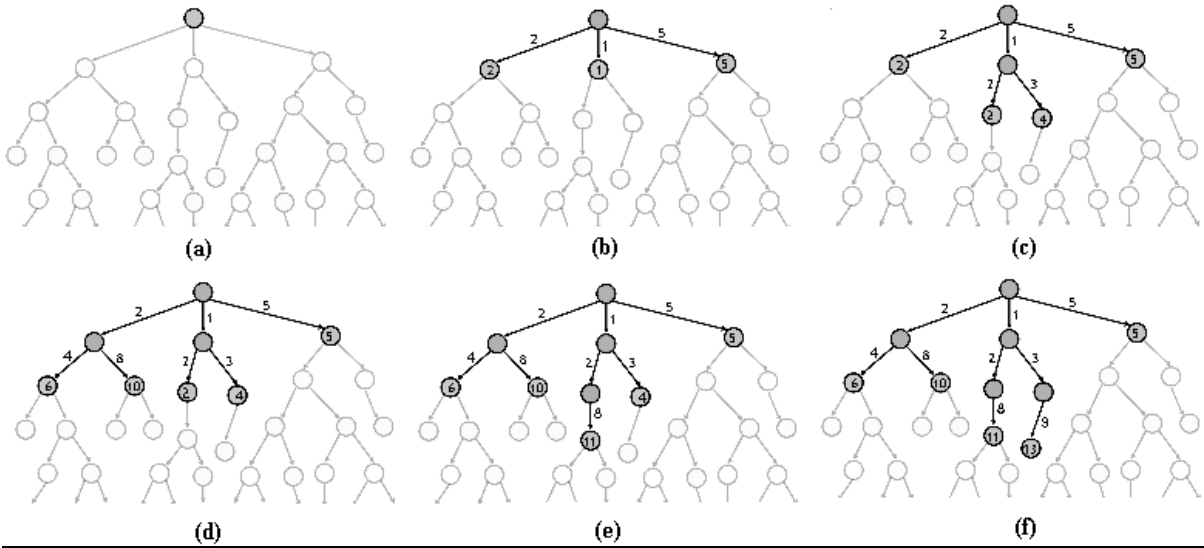

FIGURA 4.1 Busca de custo uniforme. Nos nós os números representa o custo de caminho *g(n)* e, nas arestas, os valores são o custo de passo entre um nó e outro. Na ordem de expansão de nós inicia-se pelos de menor $g(n)$, sendo esse processo ilustrado na seqüência (a), (b), (c), (d), (e) e (f).

4.4.3 Busca em profundidade (*Depth First Search*)

Na busca em profundidade, o nó mais profundo da borda inexplorada à esquerda da árvore de busca é expandido primeiro, ou seja, a busca procede até os nós folhas e, então, retrocede ao nó ainda não expandido mais profundo à esquerda, reiniciando esse processo (Russell & Norvig, 2004). Este procedimento é mostrado na figura 4.3, com as setas pontilhadas indicando a ordem de exploração de nós em uma dada árvore.

Esta estratégia permite varrer toda a árvore para chegar ao nó que satisfaça uma condição alvo. Porém, existem problemas de árvores ilimitadas, como, por exemplo, querer verificar, por exploração de nós, se uma marcação é alcançável em uma rede de Petri ilimitada, contendo transições vivas que cubram a marcação anterior. Para casos como esse, pode-se usar uma variação chamada *busca em profundidade limitada*, em que há um limite (l) de profundidade previamente estabelecido.

Eventualmente, pode-se ter um parâmetro l para o processo de busca em profundidade limitada suficientemente pequeno a ponto da solução não poder ser encontrada entre os nós abrangidos por esse limite. Estende-se, então, este método para permitir uma *busca em profundidade com aprofundamento iterativo*. Se uma solução não é encontrada dentro de um limite l de profundidade, então l é incrementado e o processo é reiniciado, garantindo assim a completude do método.

Em todos os três casos, apresenta-se uma complexidade exponencial no tempo, $\mathbf{O}(b^x)$, e de apenas $\mathbf{O}(bx)$ no espaço, em que, dependendo da busca ser em profundidade, em profundidade limitada ou com aprofundamento iterativo, x vale, respectivamente, m, l ou d.

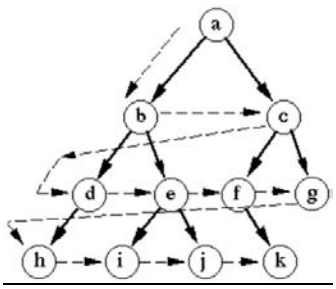

FIGURA 4.2 Busca em largura.

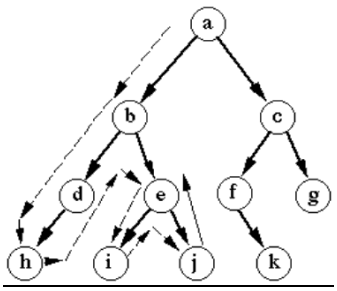

FIGURA 4.3 Busca em profundidade.

4.4.4 Busca bidirecional sem informação

Dado o fato matemático de que $b^{d/2} + b^{d/2}$ é muito menor que b^d, uma estratégia com duas buscas sendo realizadas simultaneamente, uma partindo do estado inicial e a outra inversa a partir do objetivo, reduz consideravelmente o número de nós explorados. Usa-se assim uma *busca bidirecional*, em que são tratas duas árvores de busca simultaneamente usando umas das estratégias abordadas. Ao expandir um nó de uma das árvores, verifica-se a sua pertinência na borda da outra árvore e, se for o caso, encerra o processo. Uma visão esquemática desta busca é mostrada na figura 4.4, com as ramificações de ambas as árvores quase se encontrando (Russell & Norvig, 2004).

Esta abordagem é completa, se ambos os sentidos usarem busca em largura. Se, além disso, os custos de passo forem todos idênticos, ela também acha a solução ótima. A complexidade, tanto de tempo como a de espaço, é $\mathbf{O}(b^{d/2})$, como discute Russell e Norvig (2004).

As questões limitantes desta estratégia relacionam-se com a reversibilidade das ações do problema, com o número de estados objetivos possíveis e com a descrição do teste de objetivo. O primeiro aspecto refere-se à necessidade da existência de uma função que permita saber quais os possíveis nós pais de um estado. Em casos com vários estados objetivos, pode-se criar um estado alvo fictício como sendo o sucessor de todos esses nós. Outro aspecto é o fato de nó alvo nem sempre ser conhecido, podendo este estar definido pela satisfação de uma condição implícita, como o "xeque-mate" no jogo do xadrez (Russell & Norvig, 2004).

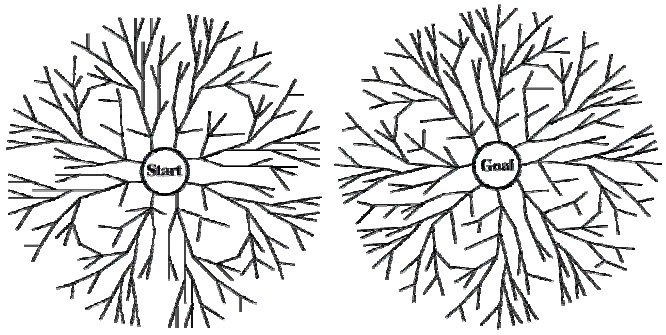

FIGURA 4.4 Visão esquemática da busca bidirecional (Russell & Norvig, 2004).

4.5 Busca heurística

Heurísticas podem ser embutidas no processo de busca para reduzir a tarefa de exploração do espaço de estados visando aumentar o seu desempenho. São informações definidas em termos de princípios, métodos ou critérios relativamente simples de tomada de decisão dentre as possibilidades de exploração de estados em um problema a ser resolvido (Pearl, 1981).

Em uma exploração de nós envolvendo essas informações, chamada de *busca heurística,* há uma função de avaliação $f(n)$ que fornece uma estimativa de qual melhor caminho a ser explorado.

A heurística pode ser uma função $h(n)$ que retorna o custo estimado do caminho mais econômico de um dado nó n até um nó objetivo e, neste caso, é chamada de *função heurística*. Em geral, ela pode ser obtida ao se relaxar certas restrições de um problema, calculando-se previamente os custos de solução para casos simplificados, ou pela experiência que se tem com o tipo de problema envolvido (Russell & Norvig, 2004). Há vários tipos de busca heurística, das quais se podem destacar as clássicas: busca gulosa e busca A*.

4.5.1 Busca gulosa (*Greedy Search*)

Uma escolha natural para a função de avaliação seria $f(n) = h(n)$. A estratégia baseada na escolha do caminho mais promissor baseado nesta função de avaliação é a chamada *busca gulosa*. Em cada passo de exploração, há a tentativa de se chegar o mais próximo possível da solução. Deixando os nós menos promissores por último, a diminuição no número de nós explorados é esperada. Porém, minimizar $h(n)$ pode permitir falsos inícios, isto é, caminhos que levam a exploração de nós desnecessários.

A busca gulosa tende a seguir um caminho até objetivo, esgotando os ramos da árvore de busca até chegar a um *dead-end*, um "beco sem saída" em que se depara com um nó folha sem sucesso na obtenção da solução, similar a busca em profundidade. Não por acaso, apresenta seus mesmos defeitos da busca em profundidade – não é ótima e nem completa. Apresenta a complexidade no tempo de $\mathbf{O}(b^m)$ para o pior caso, que pode ser reduzido por uma boa função heurística (Russell & Norvig, 2004).

4.5.2 Busca A* (*A-star*)

A busca A* (lê-se "A-estrela") é uma estratégia baseada na minimização do custo total estimado da solução. Desta forma, a função de avaliação para um dado nó n é definida por $f(n) = g(n) + h(n)$, sendo $h(n)$ a função heurística do custo estimado de n ao objetivo e $g(n)$ o custo de caminho desde o nó inicial s até n, com $g(s) = 0$. Esta função torna o procedimento completo e ótimo[2], deste que a função seja *admissível*, isto é, não superestime o custo real $Ht(n)$ para alcançar o objetivo ($f(n) \leq Ht(n)$). (Russell & Norvig, 2004) (Pearl, 1981).

A idéia, como aborda Pearl (1984), é detalhada no algoritmo 4.3. Trata-se, na verdade, de uma extensão do algoritmo de custo uniforme, sendo agora considerado o custo total estimado. Se for considerada para todo nó n a heurística $h(n) = 0$, tem-se o mesmo comportamento da busca de custo uniforme ($f(n) = g(n)$).

Além da *admissibilidade*, outra propriedade interessante que $h(n)$ pode satisfazer é a *consistência* ou *monotonicidade*. Isso ocorre quando $h(n) \leq c(n, y, n') + h(n')$, para todo n e n', tal que n' seja sucessor de n. A função de avaliação nestes termos é dita *consistente* e garante que um caminho até um dado nó é sempre o de menor custo. Toda heurística consistente é também admissível. (Russell & Norvig, 2004) (Pearl, 1981).

Algoritmo 4.3

```
UNEXPLORED := { s };  EXPLORED := { };
while ( UNEXPLORED ≠ { } ) do
  begin
    q' :=  remove_from_UNEXPLORED_the_minimum_f(n)_node;
    if ( q' = <objetivo> )
       then stop_and_return_solution ( q');
       else  begin
                Q := successors_of ( q');
                for each t ∈ Q
                  begin
                     if (t ∈ UNEXPLORED)
                      then preserve_the_minimum_cost_node;
                     f(t) := g(t) + h(t);
                  end;
                UNEXPLORED := Q ∪ UNEXPLORED;
                EXPLORED := EXPLORED ∪ { q'};
             end;
  end;
```

[2] Considera-se, para isso, o número de sucessores por nó finito e os custos de passo $c(q, y, q')$ finitos e positivos.

Segue a demonstração da *optimalidade* do algoritmo de busca A* com heurística admissível ($f(n) \leq Ht(n)$):

Hipótese: supõe-se existir a possibilidade da primeira solução encontrada por A* *não* ser a ótima.

1) Seja q o estado objetivo ótimo, com o mínimo custo de caminho Ht^* e seja q_2 um estado objetivo não-ótimo, ou seja, $g(q_2) > Ht^*$

2) Supondo que A* tenha selecionado q_2 em sua fronteira de expansão sem passar por q e, como q_2 foi escolhida antes de q, tem-se que $f(q) \geq f(q_2)$.

3) Como $f(q) = Ht^*$, da desigualdade do item anterior tem-se que $Ht^* \geq f(q_2)$.

4) Por outro lado, $f(q_2) = g(q_2) + h(q_2)$;

5) Como q_2 é nó objetivo, tem-se que $h(q_2) = 0$ e, portanto, $f(q_2) = g(q_2)$;

4) Mas $Ht^* \geq f(q_2)$, o que resulta em $g(q_2) \leq Ht^*$. Isso contradiz uma condição do item 1 ($g(q_2) > Ht^*$), ou seja, conclui que o custo de um estado objetivo não-ótimo é menor ou igual ao mínimo custo possível na obtenção da solução. A hipótese dada é assim refutada pela *prova por absurdo*, comprovando-se, assim, a optimalidade de A*.

O algoritmo de busca A*, dada uma heurística admissível, é *otimamente eficiente*, isto é, dos algoritmos ótimos de busca por expansão de nós, não existe outro que necessite da expansão de um menor número nós para se chegar a uma solução.

Entretanto, o crescimento do número de nós explorados poder se dar ainda de forma exponencial em relação ao comprimento da solução. Entretanto, uma boa heurística permite se chegar a um crescimento sub-exponencial desde que o erro em $h(n)$ não cresça numa rapidez maior que o logaritmo do custo real $h^*(n)$, isto é, $\left|h(n) - h^*(n)\right| \leq \mathrm{O}(\log h^*(n))$ (Russell & Norvig, 2004).

Algumas variações permitem diminuir o número de nós em memória, comprometendo o tempo com operações adicionais, além de deixarem de ser completas ou ótimas. Por exemplo, o SMA* é similar ao A* até completar toda memória disponível e, então, um novo

nó só poderá ser adicionado se um outro nó da árvore for descartado (sempre o "pior" entre os mais profundos).

Outras variações introduzem decisões irrevogáveis para reduzir o tempo de busca, também descartando o caráter ótimo e a completude. Como, por exemplo, o DWS (Dynamic Windows Search) desenvolvido por Reyes *et al.* (2000), que restringe o espaço de busca baseando-se na idéia de uma "janela" de tamanho fixo que desliza sobre a árvore, sendo ignorados os nós fora de suas delimitações.

4.5.2.1 Função heurística

Uma forma de se avaliar uma função heurística é usar o *fator de ramificação efetiva*, b^*. Supondo que o resultado de uma busca A* é alcançar um nó objetivo a uma profundidade d e sendo, para isso, necessário chegar ao montante de N nós gerados. O fator b^* é a razão da progressão geométrica dos $d + 1$ primeiros termos, começando por 1, referentes a uma ramificação uniforme hipotética da busca, cuja soma resulta na totalidade dos nós pesquisados da árvore de busca $N + 1$. Dependendo de quão próximo de 1 for b^*, melhor será a heurística.

$$N + 1 = 1 + b^* + (b^*)^2 + ... + (b^*)^d$$

ou

$$N + 1 = \frac{1 - (b^*)^{d+1}}{1 - b^*}$$

Diz-se que uma heurística $h_2(n)$ *domina* outra $h_1(n)$ se $h_2(n) \geq h_1(n)$ para todo nó n. A busca A* com a heurística $h_2(n)$ explora em média menos nós em relação a que usa a heurística $h_1(n)$. É possível demonstrar que $h_2(n)$ nunca é pior que $h_1(n)$. É correto afirmar que A* expande todos os nós com $f(n) < Ht^*$ e pode-se concluir, portanto, que todo nó com $h(n) < Ht^* - g(n)$ será expandido. Porém, como $h_2(n)$, em todo nó, é pelo menos tão grande quanto $h_1(n)$, tudo o que é expandido com $h_2(n)$ é também expandido usando-se a heurística $h_1(n)$, embora o oposto não seja necessariamente verdade. Assim sendo, pode-se concluir que o uso de heurísticas com os maiores valores possíveis pode aumentar a eficiência do algoritmo, desde que não se superestime o custo ótimo real Ht^*.

Para a obtenção de uma heurística para a resolução de um dado problema, costuma-se adotar uma definição formal do mesmo, organizando as restrições relevantes do problema em um modelo descrito em linguagem formal. Então, um novo problema é enunciado, diminuindo-se as restrições, permitindo uma solução mais rápida e mais "curta". O custo de uma solução ótima para este problema mais simples passa a ser a heurística admissível para o problema original.

Nem sempre é possível identificar claramente se novas heurísticas obtidas são melhores que outras. Assim, havendo um certo número de heurísticas admissíveis h_1 ... h_m, em que não ocorre de uma dominar outra, a melhor heurística será:

$h(n) = \text{máx} \{ h_1(n), ..., h_m(n) \}$.

Esta função, que retorna o máximo valor de cada heurística disponível pra cada nó, é admissível e consistente.

Um outro método para a obtenção de heurísticas admissíveis é dividir um problema em outros menores, sendo que o custo ótimo de um deles pode ser usado como heurística do problema original.

A heurística pode, também, ser inferida por um algoritmo de aprendizagem, baseando-se nas experiências iniciais da busca. Um processo chamado *inferência indutiva*, que consiste na tentativa de aprender uma função a partir de exemplos de suas entradas e saídas, busca adivinhar qual a melhor opção de ação baseada no conhecimento anterior. Algoritmos de *aprendizagem por reforço* também podem ser aplicados neste contexto (Russell & Norvig, 2004).

4.6 Busca evolutiva

Diferentemente da forma clássica de algoritmo de busca apresentada nas seções anteriores, certas estratégias, voltadas para problemas de otimização, utilizam-se de um procedimento iterativo de modificações sucessivas de estados para obtenção de estados "melhores" de acordo com algum critério de otimização.

Eventualmente, a convergência para a solução ótima pode não ocorrer, mas não apresenta as restrições relativas à complexidade da manutenção de uma árvore de busca, como nos procedimentos ótimos. Para cada estado uma função avalia seu grau de adequação segundo algum critério – a *função objetivo* (ou de *fitness*).

Um conceito pertinente vem a ser a *topologia do espaço de estados* (ver figura 4.5). Trata-se de uma representação que associa uma posição de um estado a uma elevação, que pode ser definida pela função objetivo ou pelo custo de caminho. No caso de se usar a função objetivo, o uso da busca visa alcançar o ponto mais elevado, o *mínimo global.* Já, no caso de se usar o custo de caminho, a idéia é achar o *mínimo local*.

4.6.1 Busca da subida da encosta (*Hill Climbing Search*)

A busca subida da encosta é uma estratégia que modifica o estado corrente na tentativa de ir melhorando-o, sendo que a árvore de busca não é mantida. Em seguida, é escolhido o menor entre os sucessores *n*' de *n*, explorando-o e descartando os demais definitivamente.

A busca se torna, assim, mais rápida, mas apresenta alguns problemas, como, por exemplo, a localização de máximos locais, em que toda ação gera um estado "pior" que o corrente. Uma forma de lidar com isso seria a introdução de movimentos laterais, isto é, que não apresentam progresso (Russell & Norvig, 2004).

Entre suas variantes, tem-se a *subida da encosta estocástica*, que escolhe uma opção ao acaso entre os movimentos encosta acima, com probabilidade de seleção proporcional que varia de acordo com a declividade.

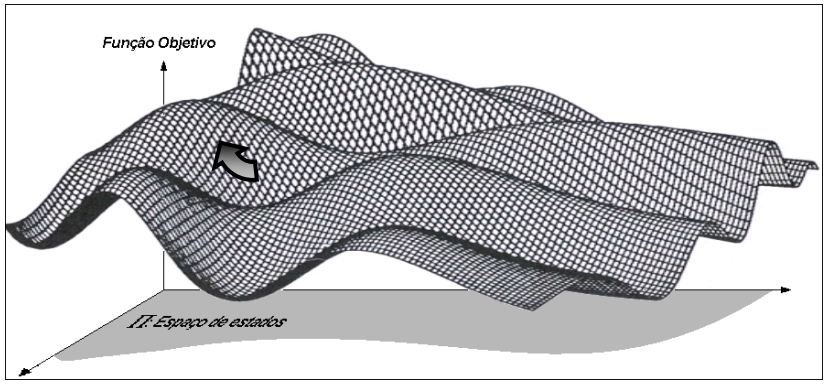

FIGURA 4.5 Topologia do espaço de estados de busca. A seta representa o caminho de uma busca da subida da encosta.

4.6.2 Busca em feixe local estocástica

Esta busca mantém o controle de *k* estados, diferentemente da subida da encosta. Em cada iteração do processo, os sucessores de cada um deles são gerados por meio de ações definidas para tal. Destes, são preservados *k* elementos, sendo os demais ignorados, de acordo com uma probabilidade que seja uma função crescente de seu valor. Essa busca apresenta certa semelhança com o processo de seleção natural (Russell & Norvig, 2004).

4.6.3 Algoritmos Genéticos

Algoritmos Genéticos (GA) é uma variação da busca de feixe estocástica, especificados primeiramente por John H. Holland (1975), em que os estados sucessores são gerados pela combinação de dois outros. A idéia do processo envolvido em GA é inspirada na teoria de seleção natural proposta por Darwin no livro "A Origem das Espécies", que descreve uma dinâmica na natureza em que somente os mais aptos sobrevivem. Desta forma, a nomenclatura usada vem da biologia, como é o caso de *cromossomos, genes, crossover* e *mutação* (Beasley *et al*., 1993) (Russell & Norvig, 2004).

Os elementos de busca são chamados de *indivíduos* e são representados por uma estrutura que é chamada de *cromossomo*. Embora na natureza um indivíduo possa ter vários cromossomos, na representação desta estratégia de busca apenas um *cromossomo* identifica um indivíduo. Mantendo a analogia, pode-se pensar em indivíduos *uni-cromossômicos*. Esses cromossomos são estruturas de dados codificadas com informações do problema e representam uma cadeia de elementos, os *genes*. Os estados do gene são chamados de *alelos*.

Começa-se o processo pela geração aleatória de *k* indivíduos, chamado esse conjunto de *população*. Uma relação avalia o grau de adequação de um indivíduo às necessidades do problema, a chamada função de *fitness*. Uma nova *geração* de indivíduos é obtida por operações genéticas. Um tipo de operação é o processo chamado *crossover*, em que há o "cruzamento" entre indivíduos, selecionando os pares de acordo com seu valor de *fitness*, sendo um indivíduo criado por troca de pedaços da estrutura dos cromossomos pais. Os k elementos de maior *fitness* são preservados e o processo se repete até estes começarem a

convergir, ou seja, os indivíduos começarem a se repetir ou a variação de *fitness* começar a não ser significativa (ver algoritmo 4.4). Outras condições de parada podem ser também estipuladas, como um número máximo de gerações.

As sucessivas gerações podem convergir para um máximo local no espectro de optimalidade. Isso pode ser evitado com o uso de uma operação genética chamada *mutação*, processo de se introduz, a cada nova geração, mudanças aleatórias em genes de uma pequena porcentagem dos indivíduos, permitindo a possibilidade de novos pontos de convergência.

Especificações de algoritmos genéticos foram desenvolvidas para uso em conjunto com redes de Petri, como é discutido no capítulo 5.

Algoritmo 4.4 (Beasley et al. 1993)

```
BEGIN __ genetic algorithm __
      generate initial population
      compute fitness of each individual
      WHILE NOT finished DO
      BEGIN __ produce new generation __
            FOR population_size _ _ DO
            BEGIN __ reproductive cycle __
                  select two individuals from old generation for mating
                  __ biassed in favour of the fitter ones __
                  recombine the two individuals to give two offspring
                  compute fitness of the two offspring
                  insert offspring in new generation
            END
            IF population has converged THEN
            finished __ TRUE
      END
END
```

CAPÍTULO 5 *Uso de redes de Petri na Programação da Produção de FMS*

ABORDAGENS de Programação da Produção de FMS têm considerado dois aspectos: a modelagem e a estratégia de solução. Um número de trabalhos vem apontando o uso de redes de Petri como uma opção para essa modelagem, pois têm permitido uma representação adequada das características e restrições naturais de um FMS, além de oferecer informações úteis para uma análise para a solução do problema. Várias técnicas foram usadas em conjunto com TPN a fim de tornar essa resolução possível.

Em algumas abordagens foram aplicadas modelagens em rede de Petri temporizada (para simplificar serão chamadas simplesmente de rede de Petri), em conjunto com alguma técnica de obtenção da programação da produção segundo algum critério. São métodos que fazem a programação na etapa de planejamento, antes da execução dos processos no chão de fábrica (*offline*). Tratam-se, pois, de métodos não-reativos em sua essência, embora haja exemplos que propõem formas de reprogramação dado a ocorrência de um distúrbio na lógica corrente da produção. Na maior parte dos casos, o *makespan* foi usado como critério de medida de desempenho, sendo explicitados os que adotam outra medida. Alguns trabalhos têm enfocando a modelagem, técnicas de busca ou a heurística. No primeiro caso, é comum a preocupação de se contemplar as características e restrições reais de FMS e evitar travamentos por espera cíclica de recursos, o chamado *deadlock*.

Lee e DiCesare (1992) propõem um método baseado em busca heurística e rede de Petri temporizada para a Programação da Produção. Uma vez modelado o sistema em rede de Petri, o problema transfere-se para a busca do caminho da melhor solução na árvore de alcançabilidade correspondente. Para tanto, um intervalo de tempo k_i é associado a cada lugar p_i da PN e o disparo de uma transição t computa o *tempo máximo remanescente* τ_{MAX} dentre os associados aos seus respectivos lugares de entrada. Esse atraso pode ser entendido como o máximo tempo de contingência das marcas antes de serem consumidas pela transição.

Cabe observar que, embora pareça razoável que o custo $c(\boldsymbol{M}, \boldsymbol{M}')$ de um arco, de uma transição t, na árvore de alcançabilidade da rede de Petri temporizada possa ser simplesmente igual ao máximo atraso k_i dentre os lugares de entrada p_i de t, esse argumento não é inteiramente verdadeiro, como assinalam Lee e DiCesare (1994), pois, quando uma marcação $\boldsymbol{M}$ é convertida para uma $\boldsymbol{M}'$, os lugares que não são entrada de t também têm os respectivos tempos alterados. Neste domínio, uma variação do algoritmo de busca heurística A*, chamada L1, foi proposta como estratégia para procurar o encadeamento de disparos de transições no

tempo, que leve ao mínimo *makespan*, ou seja, ao menor tempo total de processamento do sistema modelado.

Para isso, funções heurísticas $h(n)$ relativamente simples foram propostas e aplicadas, incluindo o caso extremo $h(n) = 0$ que torna o comportamento da estratégia similar à busca do custo uniforme (um caso de busca sem informação). Duas dessas funções não garantem o limite inferior da condição de admissibilidade, ou seja, podem ser negativas. Resultados comparativos de suas aplicações são apresentados posteriormente (Lee e DiCesare, 1994). Essa abordagem acha a solução ótima, ou quase ótima, para o *makespan*. Certas condições foram assumidas, como, por exemplo, tempos de *setup* e *transporte* "embutidos" no tempo de operação das máquinas, a não interrupção ou falha de processos iniciados e tempos de processamento determinísticos. Fluxo de materiais e *setup* são reconsiderados pelos mesmos autores em trabalhos do gênero (Lee e DiCesare, 1993a, b).

Cheng *et al.,* 1994, apresentam uma modelagem em PN de FMS envolvendo elementos tais como máquinas, *buffers* limitados, robôs e sistema de transporte (AGVs), sobre o qual aplica uma busca heurística baseada na função de Lee e DiCesare, 1994.

Chen *et al*., 1994, propõem um esquema de programação da produção dinâmica por meio de uma estrutura hierárquica baseada na modelagem de FMS em redes de Petri em conjunto com busca heurística, que permite tolerância a falhas e evita *deadlocks*. A programação da produção é feita de antemão - *offline*. No processo de fabricação, um módulo é acionado por um controlador para fazer ajustes na programação em tempo real. Se o controlador detecta a ocorrência de um imprevisto (quebra de máquina, inserção de tarefas urgentes, *deadlocks*), então é feita a reprogramação da produção por um outro módulo, sendo este também baseada em busca heurística, porém com uma estrutura de PN diferente cuja marcação inicial reflete as condições correntes após o incidente. O objetivo é obter o mínimo *makespan* em meio a distúrbios naturalmente presentes em um FMS evitando *deadlocks* por meio de um algoritmo baseado na matriz de incidência da nova PN obtida.

Abdallah *et al.*, 1998, propõem um método de busca para programação de FMS modelado nos padrões de uma classe de modelos, feitos em redes de Petri temporizadas, definida como S^4R, de Systems of Sequential Systems with Shared Resources (sistemas de sistemas seqüenciais com compartilhamento de recursos), objetivando também obter o mínimo *makespan*. Um conceito presente nos modelos S^4R está na obtenção de um

comportamento livre de travamentos por espera cíclica por recursos, a ausência dos chamados *deadlocks*. O método de busca baseia-se no princípio *branch-and-bound* (ramifica e limita) que usa regras de prioridade presentes em decisões no chão de fábrica para reduzir a árvore de estados no processo de exploração, que no caso, é feito usando conceitos da busca por retrocesso. Alguns estudos de caso foram apresentados com tempos de resposta aceitáveis, embora a complexidade do algoritmo seja exponencial.

De forma similar, buscou-se satisfazer mais de um critério na programação da produção (Yim & Lee, 1996). Diferentes funções heurísticas não admissíveis foram propostas para um processo de busca (similar a "gulosa") que considera duas delas, por média ponderada algébrica ou por média geométrica, no intuito de atender objetivos diferentes (e aparentemente contraditórios) – o mínimo *makespan* e o menor custo total de operação. As funções, entretanto, são parametrizadas por constantes que devem ser melhoradas gradualmente para se adequarem ao problema. Um certo cuidado, inclusive, faz-se necessário na questão da dimensionalidade das variáveis envolvidas na média calculada, que envolve o uso de parâmetros de ajuste para lidar com as diferentes grandezas dos números envolvidos.

Nas estratégias descritas, a avaliação do custo remanescente no processo de busca não está baseada em uma teoria analítica, considerando apenas o estado corrente ou informações globais limitadas. Em meio a essas preocupações, abordagens envolvendo soluções aproximadas de equações de estado de rede de Petri foram propostas visando obter funções heurísticas que reduzissem o número de nós explorados no processo de busca pelo mínimo *makespan*. Jeng e Chen (1995a) propuseram uma busca usando uma variante do A* com uma heurística admissível baseada em equações de estados, encontrando assim a solução ótima. Uma nova abordagem usando esta heurística foi feita para uso em uma busca sem retrocesso, aproximando-se da busca da *subida da encosta* (Jeng e Chen, 1995b). Mais tarde, Jeng e Chen, 1998, fazem um estudo mais completo desse processo, descrevendo as etapas envolvidas (algoritmo de busca ótimo, função de custo de caminho, função heurística e inserção de estados em uma lista) e fazendo uma comparação dos resultados com os trabalhos de Lee e DiCesare (1994). A idéia para a obtenção da função heurística é reduzir a equação de estados $\Delta M = C^T \mu$, obtendo $\boldsymbol{A}X = \boldsymbol{B}$, sendo $\boldsymbol{A}$ e $\boldsymbol{B}$ submatrizes obtidas das matrizes reduzidas por eliminação gaussiana de ΔM e C^T, respectivamente, $[\mathrm{A}^{\mathrm{T}}0]^{\mathrm{T}}$ e $\left[B^T B_Z^T\right]^T$. A função heurística que garante a solução ótima é dada a partir da mínima soma dos produtos dos

elementos de *X* pelos tempos de transição correspondentes. Porém, esse é um problema de programação com inteiros que envolvem passos relativamente complexos para sua resolução. O número de nós explorados é reduzido, mas uma série de passos é introduzida a cada geração de um sucessor e um espaço de memória considerável é necessário para o cálculo com matrizes. Estatísticas experimentais demonstraram que esta estratégia pode vir a ser de três a cinco vezes mais demorada que a de Lee e DiCesare (1994). Novos resultados com grande volume de dados de entrada gerados aleatoriamente ratificam isso (Jeng e Chen, 1999).

Já na abordagem de Jeng e Lin, 1997, a obtenção do modelo do FMS, sobre o qual se aplica todo esse processo, se dá pela junção de módulos em PN (que são chamados de *RCN - Resource Control Nets*), por refinamento (explosão) de *transições* do modelo em rede de Petri. Foram definidas duas classes de rede de Petri para este tipo de aplicação – redes simétricas e assimétricas. Uma rede assimétrica possui roteiros por tipos de produto, *jobs*, em que há partes com seqüências alternativas de operações, enquanto que na rede simétrica as partes possuem uma única seqüência de operações. As PNs dessas classes são limitadas e comportam-se como vivas até atingir uma marcação desejada.

O uso de equações de estados foi também aplicado para a obtenção de uma heurística visando reduzir o grau de atraso efetivo, o *tardiness*, em vez de achar o mínimo *makespan*. Para cada *job,* define-se previamente um peso. Em determinado ponto da busca, calcula-se, pelo tempo corrente, qual o *tardiness parcial*. Uma função usa equações de estado para estimar o *tardiness* das tarefas remanescentes, que é somado ao valor parcial para obter o *tardiness total estimado*. Este procedimento foi usado em um algoritmo de busca e os resultados foram comparados com métodos de despacho de serviço clássicos baseados em informações locais (FIFO, LSO, LS, EDD[1]) mostrando um desempenho melhor em relação a estes (Jeng, Jaw e Hung, 1997).

Um algoritmo de busca foi proposto por Reyes*,* Yu e Kelleher (2000) apresentando melhorias na forma como o tempo é tratado na geração de sucessores e introduzindo decisões irrevogáveis para diminuir a exploração de nós. Trata-se de uma estratégia composta por dois métodos básicos: o gerador inteligente de sucessores (*IGS*), e a busca da janela dinâmica

[1] FIFO – Fist-In-First-Out (o primeiro que entra é o primeiro a sair); EDD - Earliest Due Date (menor data de entrega); LS – Least Slack (menor "pilha"); LSO – Least Slack per Operation (menor "pilha" por operação) .

(*DWS**). Quando uma transição está habilitada, ela pode ou não disparar. Assim, o *IGS* abre duas possibilidades para exploração: o estado oriundo do disparo no momento da habilitação da transição e o estado sem essa ação. Desta forma, novos encadeamentos de ações podem conduzir à verdadeira solução ótima. Contudo, isso pode não ocorrer, pois, no intuito de evitar o crescimento exponencial do número de nós a serem explorados, o método *DWS** restringe o espaço de busca, pois, baseia-se na busca A* acrescida da idéia de uma "janela" de tamanho fixo que desliza sobre a árvore, sendo ignorados os nós externos a ela (como citado no Capítulo 4). Três funções heurísticas foram testadas nesse algoritmo e os resultados demonstraram uma maior eficiência que o de Lee e DiCesare (1994) de acordo com o método de avaliação de desempenho que os mesmos haviam propostos.

Esta idéia é aprimorada por Reyes, Yu e Kelleher (2002) pela *busca de adiantamento de estágio dinâmico* (do inglês *dynamic look-ahead stage search - DLSS*) com a introdução de novos conceitos e de um estudo de eficiência e optimalidade. Essa nova nomenclatura, bastante adequada, é usada no lugar de "janela" e gerador "inteligente" sendo, respectivamente, *quadro de busca* e *gerador controlado de sucessores*. Os resultados, baseados em diferentes testes, incluindo o método de avaliação de desempenho de Lee e DiCeasere (1994), confirmaram certo grau de optimalidade (encontrou-se, em média, um *makespan* 18,8% maior que a solução ótima) conseguindo evitar um custo exponencial de exploração de nós em relação ao tamanho do problema.

Reyes, Yu, Kelleher e Loyd (2002) apresentam um padrão para modelagem de FMS e uma subclasse de rede de Petri temporizada que são propostos para o uso em conjunto com busca heurística – a chamada rede de *Buffer* ou *B-net*. Nesta variante, os *lugares* da PN são compostos pelos conjuntos disjuntos de lugares *armazéns* e *recursos*. Mediante certas condições (condizentes com tipo de problema tratado), verifica-se que uma *B-net* é *viva, limitada* e *consistente* (Yu *et al.*, 2003a). Foi, também, proposta uma função heurística para o uso de algoritmos de busca em conjunto com modelos em *B-nets*. Para isso foi criado o conceito de *matriz de alcançabilidade de custo de recurso* (*RCR*). Um algoritmo gera, partindo desta matriz, uma função admissível que apresentou uma melhor optimalidade no algoritmo DLSS do que uma função de abordagens anteriores (Lee e DiCesare, 1994, e Yim e Lee, 1996) em 81% dos problemas testados (cerca de 1000 casos gerados aleatoriamente segundo algumas características).

Baseado nesta idéia, Yu *et al*. (2003a e 2003b) apresentaram uma forma de formalização do problema de programação de FMS, e criaram uma linguagem formal para padronizar a definição de problemas, chamada FmsML, para programação da produção diante de algumas condições assumidas. Detalhou-se, então, um procedimento para a obtenção de um modelo em *B-net* de um sistema definido em FmsML. Foi também proposto um algoritmo chamado de "busca em estágios", aparentemente uma evolução do DLSS*, que deve ser aplicado sobre um modelo em *B-net,* usando a função heurística baseada na matriz de alcançabilidade de custo do recurso.

Algumas abordagens utilizam-se de controles difusos[2] para tomada de decisões no processo de busca *offiline* da solução, como a feita por Xiong *et al*. (1995) que abstrai a idéia de regras de despacho como *tempo de fluxo* no processamento de um produto, o *grau de atraso* e o *grau de atraso efetivo*. Regras difusas são aplicadas a essas variáveis e, então, um método difuso, proposto por Tsukamoto (1979) *apud* Xiong *et al*. (1995), é usado no controle para decisões de disparo em PN. Em outro exemplo, Xu (1996) trata a possibilidade de uso do controle difuso em um FMS cujos períodos de operação de uma peça dependem consideravelmente de fatores humanos, sendo o tempo de processamento, portanto, não determinístico.

Um enfoque tem sido dado ao uso de busca por algoritmos genéticos (GA) sobre a representação do sistema em PN. Reyes, Yu e Loyd (2001) propuseram uma metodologia que integra a busca heurística baseada em PN e uma análise estrutural do grafo da PN para um processo de agrupamento de serviços e tamanhos de lotes ou ambos. Um procedimento evolutivo, envolvendo Algoritmo Genético (GA), é usado para juntar e por em seqüência sub-problemas em que o carregamento, a definição de rotas de materiais e parte do problema de seqüenciamento já foram previamente determinados.

Na abordagem de Chiu e Fu, 1997, a modelagem de FMS é feita usando PN temporizadas, sendo o modelo composto pelo *modelo de fluxo de processo* e pelo *modelo de transporte*. Sobre o modelo completo, aplica-se uma busca por GA. A representação dos nós

[2] Conjuntos difusos são aqueles cujos limites não são bem definidos, permitindo graus de incerteza sobre a pertinência de elementos a eles. A lógica difusa é uma forma de tratar o raciocínio aproximado usando expressões lógicas que descrevem a pertinência de elementos a conjuntos difusos. Controle difuso faz parte de uma metodologia para a construção de sistemas de controle cujo mapeamento de valores reais na entrada e na saída é representado por regras da lógica difusa (Russel & Norvig, 2004).

(programações) usados no GA é um conjunto de listas que estabelecem uma ordem de prioridade que resolve os pontos de conflito da rede, sobre os quais são aplicadas as operações de *crossover* e *mutação*, obtendo-se novas estruturas com novas listas de prioridades. A partir desta representação, obtém-se uma programação da produção pelo disparo sucessivo de transições no modelo obedecendo às listas de prioridades e, assim, diferentes critérios podem ser avaliados, como o mínimo *makespan*, máxima porcentagem de produtos que cumprem os prazos entre outras. Também, por meio de algoritmo genético, adota-se a reprogramação da produção incluindo outras métricas no esquema de prioridades como o WIP.

Gang e Wu (2004, 2002a, 2002b) desenvolveram um método dinâmico de programação da produção incorporando busca baseada em algoritmos genéticos. Tendo em vista a verificação da segurança de PN, o modelo é submetido a uma rotina de busca GA como uma forma de garantir que o resultado da simulação de disparos de transição seja alcançável. Todos os cromossomos testados são possíveis programações da produção que evitam a ocorrência de *deadlocks*.

CAPÍTULO 6 *Uma Heurística para a Programação de FMS*

COMO visto no capítulo anterior, um número de abordagens tem investigado a programação de FMS enfocando dois aspectos: a modelagem e a resolução do problema. Também, constatou-se que a rede de Petri tem sido apontada como uma forma conveniente para modelar com certo grau de fidelidade as principais características de um FMS. As propriedades de um modelo em PN permitem que um número de estratégias de solução possa ser aplicado para resolver o problema da programação da produção. Porém, um dos grandes desafios é obter, em um tempo viável, uma programação da produção com alta precisão na minimização de custo de produção em termos de períodos de processamento, visto que se trata de um problema de grande complexidade computacional.

Um processo de obtenção de programação da produção com tempo de resposta relativamente baixo pode viabilizar uma política de "reprogramação corretiva", em que seja possível obter uma nova programação em tempo viável, na ocorrência de algum imprevisto, tal como uma falha em uma máquina do chão de fábrica, por exemplo.

O uso de uma função heurística em conjunto com o processo de exploração de estados de um FMS modelado em PN pode reduzir o tempo de obtenção de uma programação eficaz por meio de algoritmos de busca baseados na exploração de estados em árvore. Há, pelo menos, três formas para se tentar alcançar este objetivo: (1) simplificando o modelo, limitando certas características do sistema durante a modelagem; (2) usando uma busca eficiente, que, em geral, descarta ramos da árvore de estados que, possivelmente, poderiam levar à solução *ótima*, ou seja, eventualmente descartando o caminho de busca que poderia levar a uma programação que implique no mínimo custo de produção; (3) usando uma heurística eficiente, cujo uso permita que um processo de busca possa investigar primeiro os ramos da árvore de estados que mais provavelmente levarão à melhor solução possível, reduzindo o tempo de obtenção de uma resposta.

O uso de uma abordagem híbrida, na conjugação de características que abrangem os três métodos descritos, pode vir ser uma interessante solução para a obtenção de uma programação de FMS em um tempo viável. No entanto, uma certa cautela faz-se necessária neste tipo de abordagem para manter uma aceitável acuidade da solução, pois a eficácia da mesma estará justamente limitada (1) pelo grau de simplificação do problema, (2) pelas decisões de redução do espaço de estados para busca e (3) pela proximidade da heurística

(uma estimativa) em relação ao custo de produção mínimo remanescente que se é possível no sistema modelado.

Neste contexto, este trabalho enfoca a última das três opções, propondo uma nova heurística na tentativa de se obter uma programação da produção de um FMS modelado em PN, visando obter o mínimo *makespan*.

Primeiramente, descreve-se uma definição para o problema de programação de FMS, em que certas condições são assumidas para obtenção de uma enxuta modelagem em rede de Petri, visando reduzir o espaço de estados e garantir a ausência de *deadlocks*. Feitas as considerações pertinentes, é aplicada uma técnica de modelagem usando rede de Petri virtual levando em conta uma função que associa um intervalo tempo a seus elementos, permitindo a representação das durações de tempos de operação na dinâmica de transição de estados da PN. Sobre a árvore de alcançabilidade deste modelo em PN, pode-se aplicar um algoritmo de busca, para identificar o melhor caminho que atinge a marcação referente ao estado final desejado.

Define-se, então, uma função heurística que estima o mínimo custo de produção remanescente, no processo de busca, para a obtenção da condição final, o estado alvo, em que todas as operações no chão de fábrica encontram-se concluídas e os produtos finalizados. O custo, no caso, pode ser entendido como o tempo corrente, contado a partir do início da primeira operação no chão de fábrica sobre os produtos envolvidos na modelagem do problema. Objetiva-se, assim, encontrar a seqüência de operações ao longo do tempo que resultará no menor tempo total de produção.

6.1 Definição do problema

O problema da programação de um FMS, como descrito no capítulo 2, envolve decisões na alocação dos recursos de produção ao longo do tempo, tal como a escolha dos roteiros de fabricação para cada lote de peças na obtenção de um tipo de produto, definindo o

momento de execução de cada etapa. O critério adotado, como descrito anteriormente, define o desempenho da produção baseado na obtenção do mínimo *makespan*.

Em um FMS, há certo número de recursos de produção (máquinas de processamento, manipulação e transporte) que, num momento de planejamento, podem ser alocados para a operação sobre materiais na obtenção de certos produtos. Em cada máquina do chão de fábrica, podem ser realizadas diferentes operações. O termo "operação" designará no presente texto, um tipo de processamento realizado por uma determinada máquina sobre uma classe de produtos, em outras palavras, representará univocamente estas três informações: o que é feito, em qual máquina e sobre qual tipo de produto.

Assim, dado um FMS, supõe-se que, em um determinado momento, a meta seja fabricar certo número de produtos. Para isso, dispõe-se de certos recursos de produção cujas operações sobre diferentes materiais, nos diferentes estágios da obtenção de um produto, precisam ser determinadas. O problema da programação da produção desse FMS, dirigida para uma minimização do custo de produção, será, no corrente trabalho, definido como a tarefa de se determinar previamente quais operações deverão ser feitas e em quais instantes, de forma que todos os produtos estejam finalizados no menor tempo possível.

6.2 Convenções adotadas

Certas condições precisam ser assumidas de forma a delinear o escopo em que estão envolvidas as questões tratadas na solução do problema de programação. As escolhas ao se fazer tais considerações influem no grau de complexidade do problema, pois permitem reduzir as possíveis condições de um sistema real para um espaço de estados finito obtido por um conjunto limitado de ações. Visa-se, assim, obter um modelo determinístico que se comporte dentro de um universo de possibilidades completamente observável, embora eventualmente possa continuar sendo intratável, em termos computacionais, no esgotamento por exaustão ou por busca cega.

Assume-se que os elementos atuantes no chão de fábrica, tais como máquinas de processamento de peças e AVGs, nunca falham. Sendo assim, assume-se que todos os recursos envolvidos na produção estão disponíveis durante todo o processo, sejam máquinas, AGVs, matéria prima ou outros recursos envolvidos.

Para simplificar a modelagem, o tempo de transporte será considerado desprezível, e serão tratados por meio de *buffers* intermediários, como se o *buffer* de saída de uma estação de processamento fosse o de entrada de outra. Pelo mesmo motivo, a capacidade de armazenamento de peças em *buffer* será considerada na presente abordagem como sendo ilimitado.

A demora na configuração de uma máquina para uma dada operação, o chamado tempo de *setup*, são considerados no cálculo do tempo de operação, que passa a ser a soma do tempo de processamento com o tempo de *setup*. Neste caso, supõe-se que os tempos de processamento de operações são previamente conhecidos.

Um recurso realiza uma operação por vez, transformando e retornando um único lote. Essa suposição também implica, entre outras coisas, a não existência de estações de montagem ou desmontagem como recursos transformadores. Assume-se, também, que uma vez iniciada uma operação, essa não é interrompida.

Com efeito, costuma-se, de forma geral, assumir que as ações no sistema são conhecidas, determinísticas e pertencentes a um conjunto finito. Embora imprevistos sempre ocorram, desconsiderá-los na programação da produção pode chegar a ser vantajoso. Primeiramente, porque certas incertezas seriam eliminadas do modelo, eximindo assim a necessidade de serem tratadas de alguma forma, sejam por métodos probabilísticos, raciocínio aproximado ou algum outro método aplicável ao caso em questão. Além disso, na abordagem por busca, limitar as ações diminui consideravelmente os possíveis estados do sistema, permitindo buscas mais eficientes, chegando a uma resposta mais rapidamente. Isso se justifica se for considerado um outro ponto da questão que é a possibilidade de se permitir políticas de decisões de controle no chão de fábrica que levam em conta a reprogramação da produção em tempo hábil, na ocorrência de algum imprevisto, como a quebra de máquinas, inserção de novos produtos entre outras coisas.

6.3 Modelagem do problema

Para a modelagem do problema, faz-se uso de rede de Petri virtual considerando os tempos de operações. Pode-se obter o modelo em PN a partir de uma outra representação formal mais simples, como descrito a seguir.

Em um FMS, podem-se ter roteiros alternativos para cada produto a ser produzido, definindo diferentes seqüências de operações possíveis na obtenção de um mesmo produto. Essas seqüências podem ser representadas por cadeias de símbolos concatenados (*strings*). Assim, um conjunto das possíveis seqüências de operações de um produto pode ser dado por um conjunto regular de *strings*, e, portanto, pode ser representado por uma expressão regular (Hopcroft, 1939).

Por exemplo, sejam as possíveis operações, para a obtenção de um produto em um FMS, dadas pelo conjunto de símbolos $\Sigma_1 = \{a, b, c, d, e, f, g\}$. Um possível conjunto dos roteiros alternativos pode ser representado por $L_1 = a(bc|de)(f|g)$, em que a disposição justaposta de símbolos representa a concatenação destes, o símbolo "barra vertical" (|) representa alternativas mutuamente exclusivas na composição da *string* e os parênteses estabelecem a precedência de operadores. Para este caso, abreviam-se as possíveis seqüências $L_1 = \{abcf, abcg, adef, adeg\}$.

A partir de uma expressão regular pode-se obter um autômato finito de estados correspondente e vice-versa (Hopcroft, 1939). Assim, tem-se uma representação do autômato em grafo, sendo os arcos rotulados com os símbolos correspondentes às operações no chão de fábrica. É possível, a partir desta representação, obter um modelo em rede de Petri do roteiro de fabricação do produto. Para isso, faz-se a correspondência dos vértices do grafo com os lugares de PN e dos arcos rotulados do grafo com o trio arco-transição-arco de PN. Nesta representação os lugares definem os diferentes estágios de fabricação de um produto, e podem ser entendidos como *buffers* intermediários para os produtos em progresso (materiais em percurso), que são representados por *tokens*. As transições representam as operações no chão de fábrica. Há lugares no modelo em PN que não possuem arcos entrantes, que representam os *buffers de partida* dos produtos em progresso, em que os *tokens* representam a matéria

prima ainda não processada pelo sistema para a obtenção dos produtos. Por outro lado, aqueles lugares que não possuem arcos de saída, representam os *buffers de chegada*, em que os respectivos *tokens* representam a produto finalizado.

Pegando o exemplo citado, $L_1 = a(bc|de)(f|g)$, têm-se o grafo do autômato de estados finito correspondente na figura 6.1. Obtém-se, assim, o modelo correspondente em PN, ilustrado na figura 6.2.

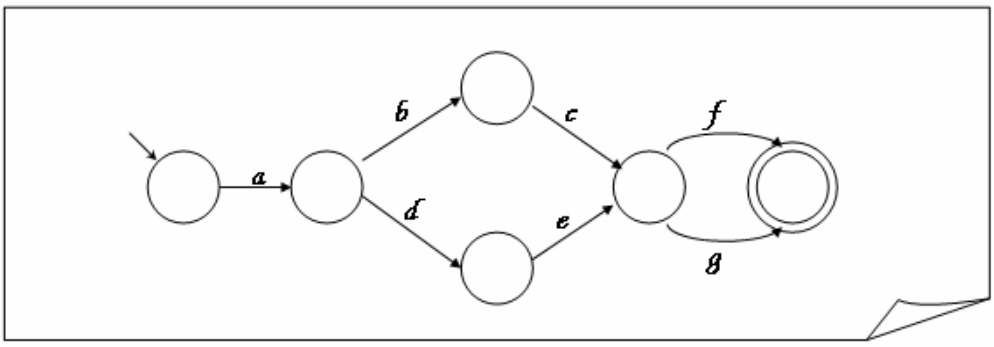

FIGURA 6.1 Grafo do autômato de estados finito correspondente a linguagem $L_1 = a(bc|de)(f|g)$.

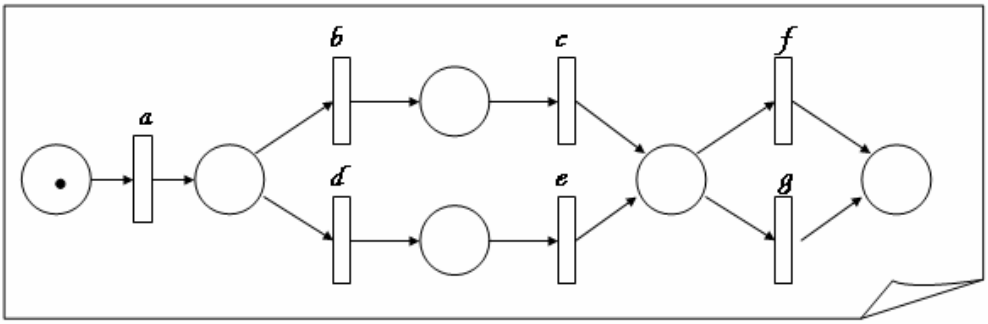

FIGURA 6.2 Modelo correspondente em rede de Petri.

Tem-se, assim, representados o roteiro de fabricação de um tipo de produto e todas as operações que possam estar evolvidas. Trata-se de uma parte do modelo, aqui chamado de *módulo*. Para cada tipo de produto a ser produzido, obtém-se o modelo correspondente segundo o processo descrito. No entanto, uma operação envolvida para a fabricação de um produto, representada por uma transição no modelo em PN, só poderá ser executada em um dado momento se o recurso de produção necessário para tanto não estiver realizando nenhuma outra operação, sobre um outro produto em progresso. A disponibilidade de recursos é o que

interliga, nesse processo proposto de modelagem, todos esses módulos referentes a roteiros de fabricação.

Para isso, faz-se uso de PN virtual, acrescentando a cada módulo os elementos virtuais referentes à disponibilidade dos finitos recursos presentes no sistema modelado. No caso, ligado a cada transição t_i, acrescenta-se o trio *arco – lugar virtual – arco*, com um *token* no *lugar virtual* representando a disponibilidade do recurso necessário para a realização da operação representada por essa transição t_i. Este processo é ilustrado na figura 6.3.

A partir dos módulos obtidos da forma descrita, obtém-se o modelo final (ver fig. 6.4), por meio da junção de elementos virtuais, como descrito no item 3.4.3 do Capítulo 3.

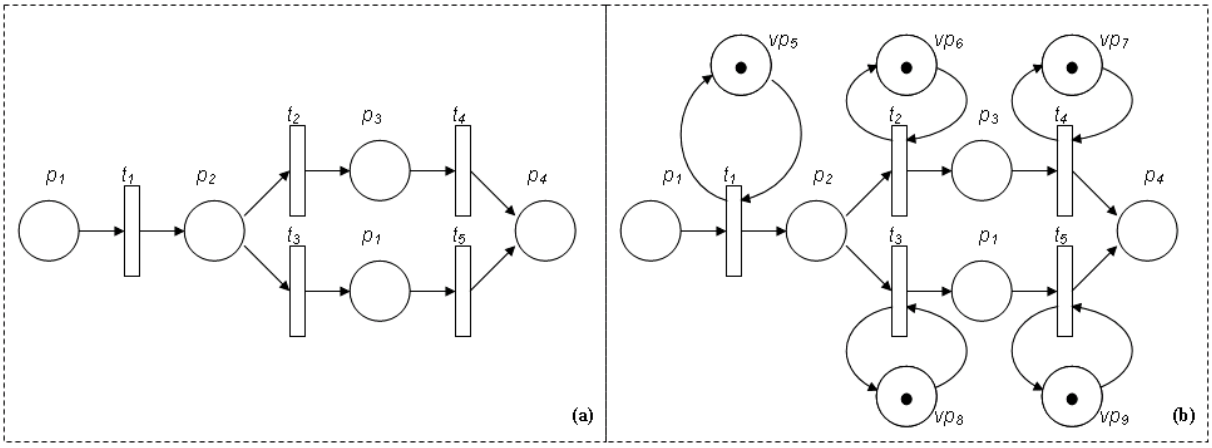

FIGURA 6.3 Inserção de elementos nos módulos para representar a disponibilidade de recursos.

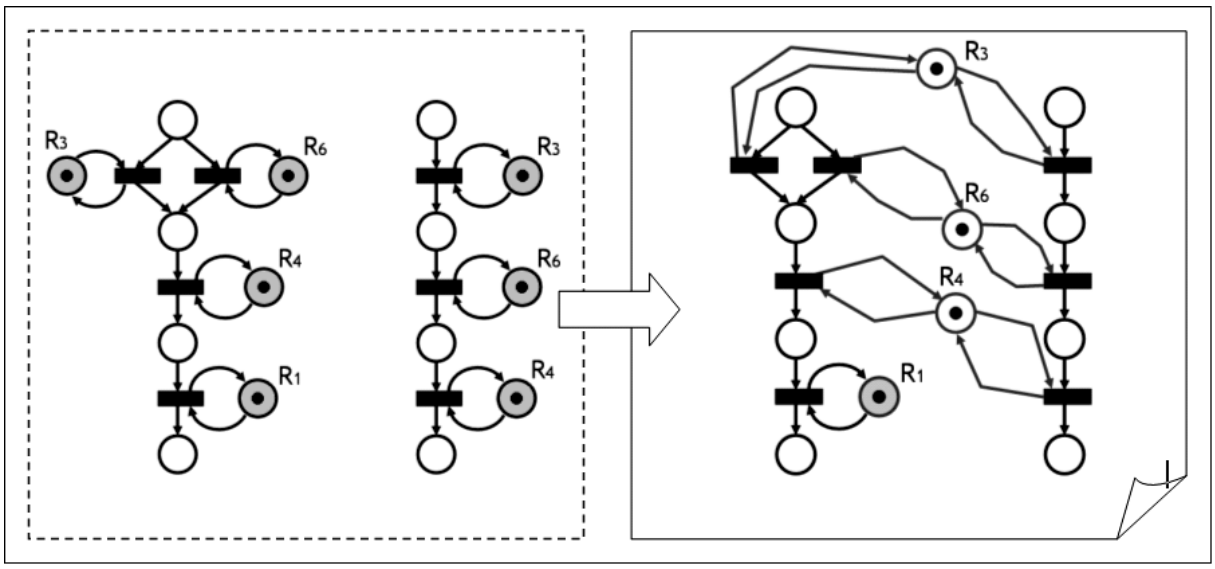

FIGURA 6.4 Junção dos módulos para a obtenção do modelo final.

O modelo em PN, assim obtido, é conservativo, uma vez que a soma dos *tokens* é constante ao longo dos disparos de transição, e livre de *deadlocks*, desde que M_0 possua, em todos os lugares que representam disponibilidade de recursos, uma marca e, em todos os lugares *buffers de partida*, uma ou mais marcas. Desta forma, a marcação de *dead-end* será somente aquela com todos os *tokens* em lugares referentes à disponibilidade de recursos e *buffers de chegada*.

6.4 Grafo de alcançabilidade e Programação de FMS

Um ramo do grafo de alcançabilidade do modelo, construído da forma descrita, representa uma seqüência de disparos de transição e, desta forma, também uma seqüência de operações no FMS. Considerando o tempo na rede de Petri, que implica uma extensão da regra de disparo, marcações temporizadas podem ser definidas. Neste caso, pode-se obter uma seqüência de disparos de transições considerando o tempo de início destes disparos.

Assim, uma busca heurística, aplicada sobre o grafo de alcançabilidade do modelo em rede de Petri temporizada (TPN), permite identificar a seqüência de operações ao longo do tempo com o menor tempo total de processamento, que é a solução para o problema da programação de FMS tal como definido anteriormente.

6.5 Busca sobre o grafo de alcançabilidade

Um algoritmo de busca pode ser aplicado sobre a árvore de alcançabilidade $R(\boldsymbol{M}_0)$, a partir do modelo em TPN, usando uma heurística que estime o mínimo custo remanescente de produção em um dado momento da produção. Os nós desta árvore, as marcações temporizadas, compõem o espaço de estados do problema. O foco desta abordagem está na heurística, ficando alheia ao escopo do corrente trabalho a discussão de qual método de busca deve ou não ser usado.

No entanto, o método de busca a ser usado deve necessariamente ter suas decisões baseadas no uso de uma estimativa do tempo total de produção remanescente a partir de um

estado do sistema, além de tratar os estados do problema modelado como marcações temporizadas de TPN.

O conjunto dos possíveis estados de uma PN representado por marcações temporizadas retrata um número discreto de situações do sistema observadas em diferentes instantes dentro de um período contínuo de tempo. A partir de uma dada marcação, o disparo de transição representa o início de uma etapa no processamento de um produto. Esta ação na PN consumirá os *tokens* referentes ao insumo a ser processado e à disponibilidade do recurso que executará o processo, gerando novos *tokens* na obtenção de uma nova marcação. À marcação gerada está associado o *tempo corrente* correspondente ao momento em que se iniciou uma dada operação. Os *tokens* gerados possuem tempos a eles associados referentes ao período de duração dessa operação que se inicia no chão de fábrica neste momento observado, que será chamado de *tempo de reminiscência*.

Se, de tempos em tempos, a situação de uma fábrica for observada, o tempo de permanência de um produto inacabado em uma dada estação de processamento tente a diminuir conforme a operação envolvida vai sendo concluída. Desta forma, de um estado para outro da PN, os *tokens* que se preservarem possuirão seus tempos de reminiscência subtraídos por um valor correspondente ao salto de tempo que se dá entre o tempo corrente de um estado observado e o que se sucede. O *tempo de reminiscência* é sempre maior ou igual a zero. Para os lugares da PN que representam *buffers,* um *token* com tempo de reminiscência diferente de zero modela a situação: "há um produto que se encontra atualmente em processamento, mas que, após um determinado período (=*tempo de reminiscência*), estará neste *buffer* disponível para a próxima etapa de processamento." Por conseguinte, o tempo de reminiscência nulo implica a existência de um produto em um determinado *buffer* do sistema.

A dinâmica na obtenção de marcações a partir de um determinado estado se dá pelo disparo de uma transição, assim como numa PN não temporizada. Porém, cada disparo de transição representa o início de uma operação no chão de fábrica. A informação do momento desta ocorrência, que pode ser definida como o *tempo corrente*, no decorrer de tempo medido no sistema em si, deve estar presente na marcação temporizada. O tempo corrente de marcação corresponde a um momento do sistema, um instante a partir do início de operação

do primeiro produto modelado na fábrica, em que o estado é retratado pela respectiva marcação de PN.

Desta forma, o produto acabado é representado por um *token* com tempo de reminiscência nulo em um *lugar de chegada*.

Assim, por exemplo, em uma PN, se uma marcação $\boldsymbol{M}_2$ é gerada pelo disparo da transição t_1 a partir da marcação $\boldsymbol{M}_1$, e sejam os *tempos correntes* de marcação $Ct(\boldsymbol{M_1}) = 4$ u.t. e $Ct(\boldsymbol{M_2}) = 9$ u.t. respectivos das marcações $\boldsymbol{M}_1$ e $\boldsymbol{M}_2$ (u.t. = unidades de tempo), então o tempo de reminiscência $\tau(u)$ de um *token* u de $\boldsymbol{M}_2$ correspondendo a preservação de uma situação representada a partir de um *token* u' de $\boldsymbol{M}_1$ é dado por:

$$\tau(u) = \max (\tau(u') - (Ct(\boldsymbol{M_2}) - Ct(\boldsymbol{M_1})), 0)$$

Se, por exemplo, o tempo de reminiscência de u' for $\tau(u') = 6$ u.t., então:

$$\tau(u) = \max (6 – (9 – 4), 0) = \max (1 , 0) = 1 \text{ u.t.}$$

Os *tokens* gerados para uma marcação pelo disparo de uma transição, que não têm correspondência na marcação antecessora, passam a ter o tempo de reminiscência equivalente à duração da operação representada por esta transição. No caso de *loop*, consideram-se o *token* consumido e o *token* gerado como sendo distintos entre si e não correspondentes em marcações consecutivas.

Assim, adaptado da abordagem de Yu *et al.*, 2003, pode-se chegar a uma definição formal de marcação temporizada.

Definição 6.2 Marcação temporizada é a quíntupla $\boldsymbol{M} = (M, M_A, Ct, \sigma, U)$, em que:

- M é a marcação com os *tokens* disponíveis e indisponíveis;
- M_A é a marcação com os *tokens* disponíveis;
- Ct é o tempo corrente, momento do disparo de transição que gerou $\boldsymbol{M}$;
- σ é a seqüência de disparos de transição, com os respectivos instantes de ocorrência, na obtenção de M a partir de M_0;

- U é o conjunto dos pares ordenados (p_u, $\tau(u)$) referente aos *tokens* indisponíveis, em que
  p_u é o lugar em o *token u* se encontra,
  $\tau(u)$ é o respectivo *tempo de reminiscência* de u;

Essa dinâmica na geração de marcas temporizadas define a função de geração de sucessores no processo de busca. A escolha de qual nó priorizar na ordem de expansão, definindo qual ramo da árvore de busca deve ser investigado primeiro, pode ser feita segundo uma estimativa que é dada pela função heurística.

6.6 Heurística

A idéia básica por trás da heurística proposta relaciona-se ao conceito de paralelismo no processamento de materiais.

Um produto inacabado, em processo, quando localizado em um determinado *buffer* do chão de fábrica pode ser considerado com se estivesse em um determinado estágio da produção. Assim, os *buffers* em que os produtos em processo se encontram podem identificar o "estágio de produção" que estes se encontram. Para a simplificação do modelo, um buffer de saída de uma determinada máquina MCH_n e um buffer de entrada de uma outra máquina MCH_{n+1}, que será o passo subseqüente de um produto que passou por MCH_n, podem ser pensados como se fosse um único buffer intermediário entre as máquinas MCH_n e MCH_{n+1}. Porém, um FMS proporciona diferentes caminhos (seqüências alternativas de máquinas) de um estágio a outro, com diferentes custos em termos de tempo de processamento. Além disso, é comum haver mais de um produto em processo que, para serem finalizados, necessitam do uso dos recursos (máquinas) compartilhados no chão de fábrica.

Desta forma, pode-se definir informalmente *paralelismo perfeito* no processamento de materiais como uma condição abstrata (hipotética e utópica) em que o processamento de um produto não interfere no processamento de outro em qualquer momento. Seria o equivalente a se ter uma fábrica, idêntica a do sistema modelado, com os recursos inteiramente disponíveis

para o processamento de cada um dos produtos em progresso considerados no modelo. Neste caso, há uma situação hipotética de um sistema livre de conflitos em que a decisão no uso de máquinas para um determinado estágio torna-se bastante simples. Escolhe-se, para um produto *i*, por exemplo, o caminho menos custoso em termos de tempo de processamento, ou seja, a seqüência de máquinas cuja soma dos respectivos tempos de processamento é mínima para o produto *i* em questão, que será referido por *flowtime ideal* (f_i^*).

Deste modo, o maior dentre os *flowtimes* ideais (f_{MAX}^*) seria exatamente o mínimo makespan em um sistema com *paralelismo perfeito*. Para um sistema com conflitos no uso de recursos, que se aproxima mais da realidade, f_{MAX}^* corresponde a um período tempo do qual não é possível obter um *makespan* com menor valor.

Mas, quando se considera um estado do sistema, têm-se diferentes produtos em diferentes estágios de produção. Os estágios dos produtos em um determinado estado de produção podem ser considerados os estágios iniciais para a obtenção de um *flowtime ideal*. Assim, na suposição do *paralelismo perfeito*, pode-se definir *flowtime ideal remanescente* $\phi_i(n)$ como sendo o caminho de menor custo na produção do produto *i*, a partir do estado *n*. Assim sendo, pode-se formular uma heurística simples para estimar o custo total de produção remanescente a partir de um certo estado *n*:

$$h_1(n) = \max\left(\phi_1(n), \phi_2(n), ..., \phi_i(n), ...\right) = \phi_{MAX}(n) \qquad (6.1)$$

Seja o mínimo custo total remanescente, considerando a concorrência entre processos no chão de fábrica, como sendo $Ht(n)$. Assim, uma heurística $h(n)$ é *admissível* se $0 \leq h(n) \leq Ht(n)$, isto é, existe um processo de busca que, quando utiliza esta heurística, é capaz de obter o mínimo custo total de produção do sistema modelado. Diz, por isso, que, pelo uso desta heurística, é possível achar a *solução ótima*, isto é, aquela que otimiza uma característica alvo que, no caso, é o tempo de produção.

Pelo fato de se basear em uma situação que pressupõe uma condição ideal hipotética não comum a FMSs, a heurística $h_1(n)$, da expressão (6.1), subestima o custo mínimo total do sistema. É fato que, para o sistema modelado, diante de certas condições, como a suposição

do *paralelismo perfeito*, o mínimo custo remanescente pode até ser igual a $h_1(n)$, mas nunca menor. Logo, é admissível. Porém, uma heurística admissível $h(n)$ é tão eficiente o quão próxima for de $Ht(n)$. Por exemplo, no caso do extremo oposto, com $h(n) = 0$, seu comportamento se assemelhará a de um método de busca não informada: a busca de custo uniforme (Pearl, 1964). Como a heurística $h_1(n)$ subestima bastante o custo mínimo total, seu uso em algoritmos de busca em árvore de estados leva a uma ineficiência em termos de tempo de resposta na obtenção de uma programação da produção.

Propõe-se, então, acrescentar à expressão (6.1) um termo que leva em conta o conflito nas decisões de uso compartilhado de máquinas. Como o custo se baseia no tempo λ de operação em máquina em um certo estágio, uma nova heurística poderia ser:

$$h_2(n) = \overline{F}_{MAX}(n) + K(n) \cdot \lambda_{DEF} \qquad (6.2)$$

em que:

- $K(n)$ é o *fator de conflito* que depende do estado n e que representa o conflito causado pela concorrência no uso de recursos a partir de n;

- λ_{DEF} é *o tempo pivô de operação* que dependente das características do sistema. Assume-se, por simplificação, que é o mínimo dentre os tempos de processamento dentre as operações remanescentes (λ_{MIN});

Para um estado do sistema n, seja $Rr_i(n)$ o número de máquinas pelas quais um certo produto em processo i ainda precisa passar. Se um FMS tem um número total de máquinas Rt, é possível definir $K(n)$ pela expressão:

$$K(n) = \frac{\sum_i Rr_i(n)}{Rt} \qquad (6.3)$$

Em um determinado estado n de produção, há a necessidade de se realizar um certo número de operações, que corresponde a $\sum_i Rr_i(n)$, para que se possa finalizar toda a

produção. Para tanto, há no sistema *Rt* máquinas. O *fator de conflito K*(*n*) é uma relação, um valor adimensional que leva em conta toda esta informação.

Entretanto, para o uso de *K*(*n*) na expressão (6.2), os *Rrs*(*n*) recursos ainda necessários para a finalização do produto de maior *flowtime ideal remanescente* são descontados do somatório, pelo fato de que estes recursos já são considerados no cálculo do membro $\phi_{MAX}(n)$.

Assim, a heurística proposta pode ser reescrita como:

$$h_2(n) = \phi_{MAX}(n) + \left(\frac{\sum_i Rr_i(n) - Rrs(n)}{Rt} \right) \cdot \lambda_{MIN} \qquad (6.4)$$

Os termos $Rr_i(n)$, *Rt* e λ_{MIN}, podem ser obtidos a partir da definição do problema, que inclui questões desde o layout do FMS até roteiros de fabricação dos produtos envolvidos.

Reyes *et al.,* 2002, mostraram ser possível obter $\phi_{MAX}(n)$ em tempo polinomial por meio da obtenção de uma matriz chamada de alcançabilidade de custo de recurso (RCR – Resource Cost Reachability).

Segue uma experimentação computacional para exemplificação do uso desta heurística.

CAPÍTULO 7 *Experimentação Computacional*

PARA testar a proposta, foi realizada a implementação de um sistema que usa a busca A* e a função heurística apresentada para fazer a programação de um FMS modelado em rede de Petri, obtendo o mínimo *makespan*.

Foram implementadas, além da heurística proposta, outras três variações de uma heurística apresentada por Reyes *et al.*, 2002, e Yu *et al*., 2003, para comparações (ver Apêndice A). No corrente trabalho elas foram apelidadas de "Reyes", "Yu" e "RCR Cost".

O sistema foi programado para gerar automaticamente um conjunto de entradas geradas por função pseudo-randômica, cada qual correspondendo a um cenário com um conjunto de produtos, com as respectivas etapas de produção, a ser produzido em uma fábrica de seis máquinas. Para cada entrada, quatro buscas heurísticas, do tipo A*, são realizadas, cada qual usando uma das quatro heurísticas implementadas.

A programação do FMS é retornada para cada entrada testada na forma de um conjunto de pares ordenados, com número da operação e tempo de início de execução, e por meio de um gráfico de Gantt. Também são apresentadas informações de desempenho comparado das buscas heurísticas, como o tempo de resposta do sistema e *makespan* encontrado.

7.1 Ambiente modelado

O FMS usado para a modelagem é o sistema mostrado na figura 7.1. Trata-se da terceira fase na concepção de uma fábrica inteligente, estudada por um grupo de pesquisa do Laboratório de Inteligência Artificial e Automação (LIAA) da UFSCar, cuja linha de pesquisa envolve aspectos do controle e automação do chão de fábrica e de soluções envolvendo PCP de sistemas automatizados de manufatura, no qual este trabalho se insere.

A fábrica é composta por seis estações de processamento, três AGVs, estações de carga e descarga e uma de manutenção de veículos. As estações de processamento são os recursos concorrentes no processamento de lotes de peças na obtenção de um número de tipos de produtos.

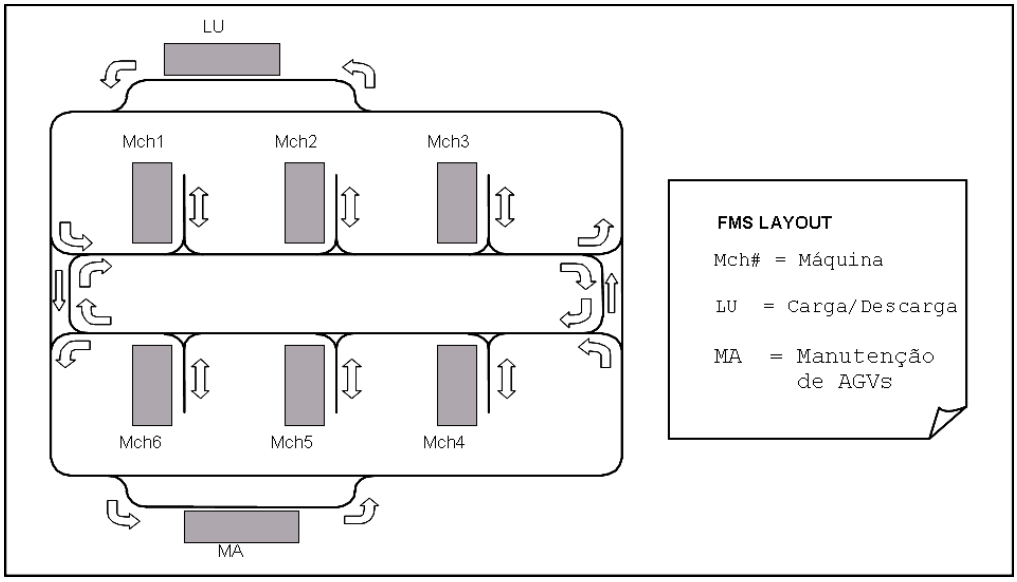

FIGURA 7.1 *Layout* do FMS modelado (adaptado de Morandin, 1999).

7.2 Processo de Modelagem

O processo de modelagem é o descrito no item 6.3 do Capítulo 6, em que se seguem os passos: (1) criação de expressões regulares que representem os roteiros de fabricação de cada produto; (2) geração de grafos etiquetados a partir das expressões; (3) obtenção de redes de Petri a partir dos grafos e (4) junção das redes de Petri em uma por meio do uso de módulos de rede de petri Virtual.

7.3 Sistema Implementado

Seguem as características e funcionalidades do sistema implementado para a realização da experimentação computacional da proposta na forma como foi previamente descrita. Também, explicita-se a forma como são geradas as entradas do sistema e a interface com o usuário.

7.3.1 Características e funcionalidades

O sistema é composto de duas aplicações implementadas na linguagem Object Pascal (usando o Borland Delphi™ 7) para serem executadas no ambiente Microsoft Windows™.

Uma delas gera automaticamente um conjunto de entradas para o sistema e, em seguida, e testa para cada entrada o comportamento de cada heurística em uma busca A*.

A segunda aplicação, permite a visualização dos resultados obtidos em gráficos e informações. Para cada busca um gráfico de Gantt informa qual a Programação da Produção encontrada. Neste texto, dá-se a denominação de "experimento" ao agrupamento de quatro buscas realizadas sobre uma dada entrada. Em cada experimento é possível visualizar graficamente o desempenho das heurísticas em termos de *makespan* encontrado e tempo de resposta. Outros dados técnicos também estão disponíveis nesta aplicação, como a profundidade das buscas, o número de nós explorados, as redes de Petri geradas entre outras coisas.

7.3.2 Geração de Entradas

A geração das entradas foi feita usando uma gramática formal de geração de expressões regulares da forma como descrita no item 6.3 do Capítulo 6. Embora as expressões sejam regulares, nada impede o uso de gramáticas de linguagens não regulares.

No Capítulo 6 expressões regulares foram apresentadas como uma forma de abreviar um conjunto de cadeias que representam roteiros de fabricação. Pode-se dizer, portanto, que as expressões regulares representam uma linguagem composta por todas essas cadeias.

Como o sistema precisa gerar essas expressões, pode-se imaginá-las não como uma linguagem, e sim uma cadeia. Assim, tudo que se faz necessário é um conjunto de cadeias de caracteres.

Para gerar esses elementos, usou-se no contexto dos aspectos formais de computação (Hopcroft, 1939), uma gramática *Gram* = ({*S*}, {*a, b, c ... z,* **(,), |**}, *P*, *S*) em que:

{*S*} é um conjunto de símbolos não terminais, no caso unitário;

{*a, b, c ... z,* **(,), |**} é um conjunto finito de símbolos terminais;

S é o elemento terminal inicial;

P é um conjunto constituído das seguintes regras de produção:

$$S \rightarrow \mathbf{(}\ S\ \mathbf{|}\ S\ \mathbf{)} \qquad (1)$$

$$S \rightarrow S\,S \qquad (2)$$

$$S \rightarrow \mathbf{a} \mid \mathbf{b} \mid \mathbf{c} \mid \ldots \mathbf{z} \qquad (3)$$

Os símbolos **|**, **(** e **)** estão estilizados para representarem elementos terminais. Já a barra vertical | é um operador da regra de produção que separa as alternativas para a geração da cadeia de caracteres. Esta distinção dos símbolos feita nas regras de produção para evitar ambigüidades na interpretação das mesmas se faz relevante apenas dentro deste contexto. O sinal de trema na terceira regra de produção, assim como na representação do conjunto de símbolos terminais, representa a continuação do padrão descrito para as demais letras do alfabeto até se chegar em **z**.

A gramática descrita permite a geração de cadeias na forma que o sistema precisa processar para gerar as redes de Petri, por exemplo: *a(b|c)de, (a|b)(c|d), (a|(c|d))e(f|g)* etc.

Para exemplificar o processo de obtenção dessas cadeias, segue uma seqüência possível na obtenção de *a(b|c)de,* usando as regras de produção acima partindo do símbolo não terminal inicial *S*:

$$S \xrightarrow{(2)} SS \xrightarrow{(3)} aS \xrightarrow{(2)} aSS \xrightarrow{(1)} a(S|S)S \xrightarrow{(3)} a(b|S)S$$
$$\xrightarrow{(3)} a(b|c)S \xrightarrow{(2)} a(b|c)SS \xrightarrow{(3)} a(b|c)dS \xrightarrow{(3)} a(b|c)de$$

A partir das cadeias geradas obtém-se a rede de Petri pelo processo modelagem descrito no item 7.2 e no item 6.3 do Capítulo 6.

Para cada rede de Petri obtida, as buscas sobre os respectivos grafos de alcançabilidade são realizadas na forma como descrita no item 6.4 do Capítulo 6.

7.3.3 Interface

Quando iniciada a primeira aplicação do sistema, uma tela surge como apresentada na figura 7.2.

FIGURA 7.2 Tela da primeira aplicação do sistema.

É possível ajustar alguns parâmetros dos cenários automaticamente gerados, como o número de máquinas no chão de fábrica, o número de produtos que se deseja produzir. Escolhe-se um nome para o arquivo gerado e a quantidade de experimentos a serem realizados. Após isso, clica-se sobre o botão "Gera Experimentos" e o processo se inicia. O decorrer do processo pode ser acompanhado por meio de uma barra de progresso que indica a porcentagem dos experimentos gerados e, posteriormente, a porcentagem dos testes realizados.

Por meio da segunda aplicação, pode-se observar os resultados para cada experimento, como é ilustrado nas figuras de 7.3 a 7.7. Por meio de um lista de no canto esquerdo, pode-se selecionar o experimento que se deseja conhecer os dados obtidos. Para cada experimento, há do lado direito um conjunto de "folhas" separadas com abas rotuladas:

- *Gantt*: mostra o gráfico de Gantt da Programação da Produção de cada busca realizada no experimento (ver figura 7.3).

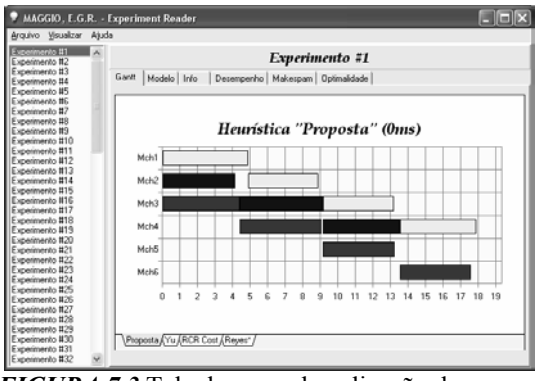

FIGURA 7.3 Tela da segunda aplicação do sistema, na aba *Gantt*.

- *Modelo*: mostras as expressões regulares geradas e a respectiva rede de Petri descrita na forma analítica (ver fig. 7.4);

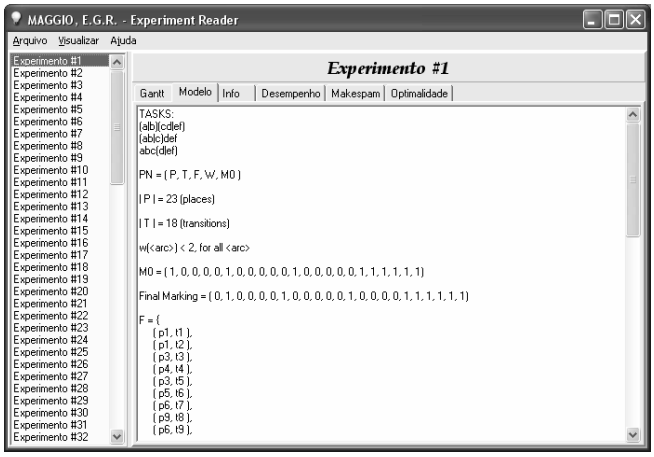

FIGURA 7.4 Tela da segunda aplicação do sistema, na aba *Modelo*.

- *Info*: mostra informações técnicas de cada busca realizada, tais como duração, número de nós explorados, makespan encontrado, profundidade de busca e as operações agendadas com os respectivos tempos de início, de duração, uso de máquina e produto a que se aplicam (ver fig. 7.5);

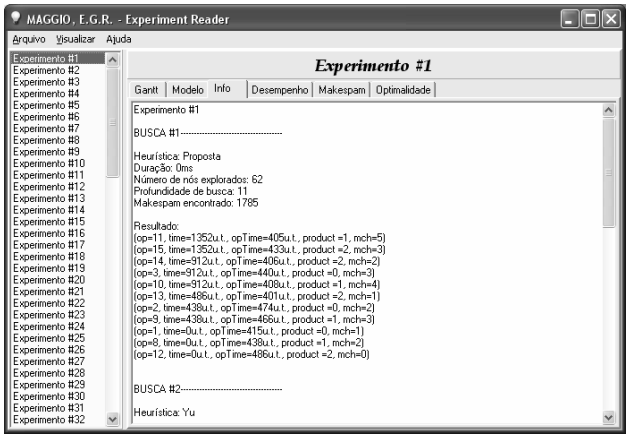

FIGURA 7.5 Tela da segunda aplicação do sistema, aba *Info*.

- *Desempenho*: mostra um gráfico comparativo dos desempenhos das heurísticas no experimento em termos de tempo de resposta (ver fig.7.6);

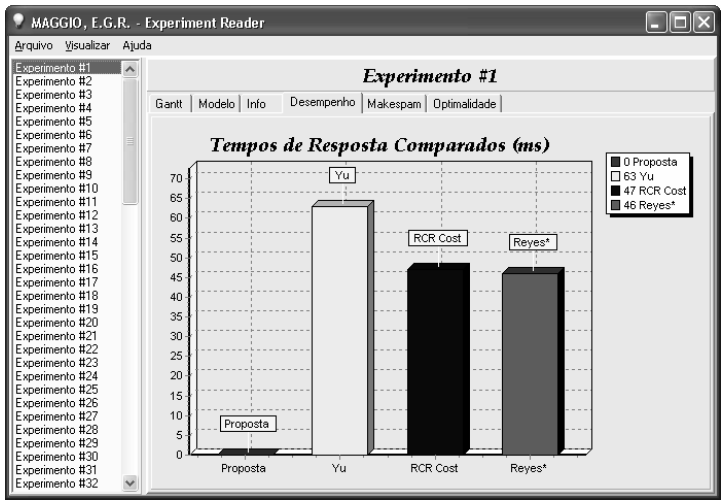

FIGURA 7.6 Tela da segunda aplicação do sistema, aba *Desempenho*.

- *Makespan*: mostra um gráfico comparativo dos desempenhos das heurísticas no experimento em termos do *makespan* encontrado (ver fig. 7.7);

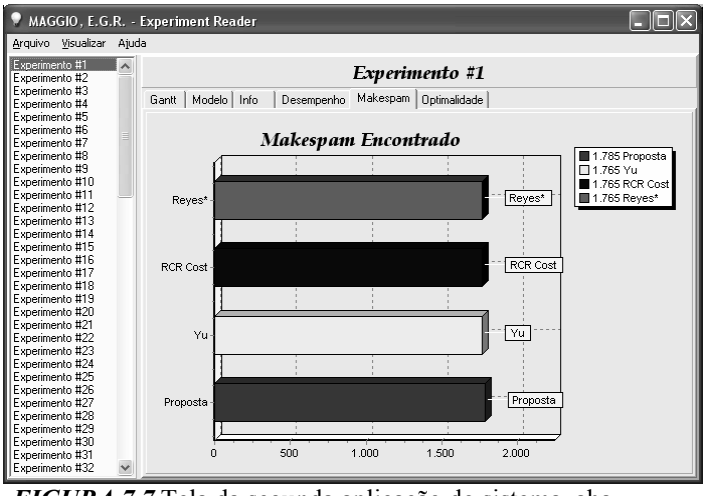

FIGURA 7.7 Tela da segunda aplicação do sistema, aba *Makespan*.

- *Optimalidade*: mostra um gráfico comparativo dos desempenhos das heurísticas no experimento em termos do *makespan* encontrado em termos relativos.

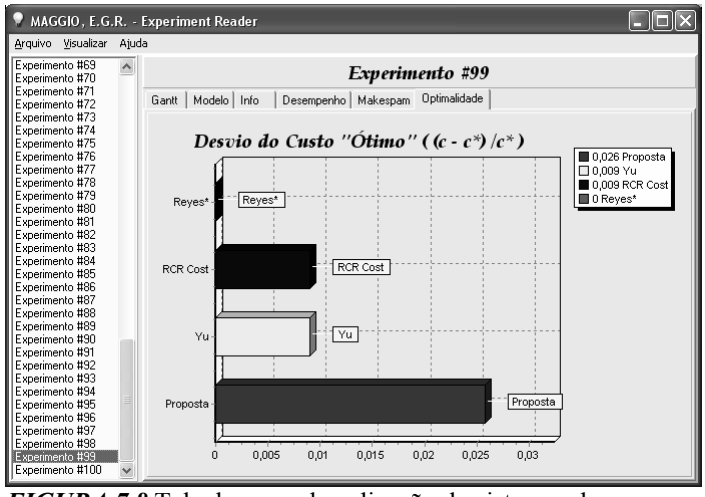

FIGURA 7.8 Tela da segunda aplicação do sistema, aba *Optimalidade.*.

Há também a opção de se visualizar todos os resultados em uma única tabela gerada automaticamente pela aplicação e compatível com o Microsoft Excel ™. Para isso, deve-se clicar sobre a opção "Planilha...". do menu "Visualizar". Para usar esse recurso, é necessário que o Microsoft Excel esteja previamente instalado no micro em que a aplicação está sendo executada.

7.4 Experimentos e Análise dos Resultados

Foram realizados cem experimentos na forma como descrita neste capítulo. Os primeiros vinte oito resultados estão listados na tabela 7.1. Uma lista completa pode ser encontrada no Apêndice B.

Em 99% dos casos, a heurística proposta manteve um desempenho, em termos de tempo de resposta do sistema, superior o das heurísticas comparadas.

Analisando a distribuição dos tempos de resposta das quatro heurísticas ao longo do tempo em intervalos de 20ms, para os valores inferiores a 1.080ms, obtêm-se as curvas de freqüência mostradas no gráfico da figura 7.4. Observa-se que as três curvas referentes às heurísticas variantes de trabalho anteriores têm maior dispersão. Em comparação a essas, o resultado a partir da heurística proposta concentrou as ocorrências próximas de zero,

apresentando uma curva de freqüência longitudinalmente estendida, mais delgada na horizontal e deslocada a direita. Na verdade, 95% dos experimentos envolvendo a heurística proposta deram um tempo de resposta inferior a vinte milisegundos, o que demonstra que a curva é mais delgada e à esquerda do que o gráfico pode levar a crer e, portanto, estando as ocorrências mais concentradas e próximas de zero.

TABELA 7.1 Resultados obtidos nos experimentos até 28º.

	Proposta			Yu			RCR Cost			Reyes*		
#	T.Resp.(ms)	Makespam	Nr. Nós	T.Resp.(ms)	Makespam	Nr. Nós	T.Resp.(ms)	Makespam	Nr. Nós	T.Resp.(ms)	Makespam	Nr. Nós
1	0	1785	62	63	1765	3653	47	1765	3896	46	1765	4059
2	0	2262	48	2218	2262	24263	2781	2262	26481	1702	2262	22183
3	0	1820	138	139	1820	6633	204	1820	8810	203	1820	8219
4	0	2354	107	78	2360	4626	343	2360	11631	391	2354	12068
5	0	2870	174	77313	2839	98185	51593	2839	79395	71968	2839	93796
6	0	1826	341	32	1802	1955	186	1802	7870	171	1802	7460
7	0	1799	76	78	1731	3863	437	1731	12174	343	1731	10897
8	0	2817	174	157	2805	6202	592	2805	14486	579	2805	14307
9	0	2780	143	8719	2715	36955	10265	2703	39822	11265	2703	40904
10	0	1845	125	219	1883	8607	156	1883	6977	140	1883	6856
11	0	2206	549	124	2272	5911	1609	2181	20681	1703	2206	20926
12	15	3265	950	200108	3224	159555	239000	3224	170037	204031	3224	155595
13	15	2383	240	31	2269	2584	140	2269	6185	203	2269	8061
14	16	2715	679	3889	2715	27262	5875	2715	32278	5344	2715	31240
15	15	2331	466	281	2238	9468	531	2238	13177	391	2238	11296
16	874	3077	16207	63125	2794	90203	57797	2794	86916	47468	2794	80327
17	0	1836	203	920	1804	15960	2079	1804	21885	1764	1804	20321
18	15	2708	968	63375	2679	86052	81969	2679	99997	57609	2679	82773
19	15	2756	1005	1889	2710	20595	17610	2710	48864	18405	2684	50144
20	15	1841	85	0	1812	796	46	1812	2691	46	1807	2668
21	0	1835	131	63	1876	3114	264	1876	9616	249	1835	9129
22	0	2384	91	124	2384	5755	108	2384	5419	125	2384	5821
23	0	2223	477	312	2223	9991	5281	2247	31341	4483	2223	29338
24	15	2321	1687	15610	2237	46581	18092	2237	49476	18312	2237	44069
25	0	2413	64	93	2406	4648	249	2373	9155	218	2373	8834
26	0	2378	209	249	2316	9144	311	2316	10396	374	2339	11531
27	0	2413	132	297	2407	9755	186	2407	8154	219	2407	8425
28	0	2322	148	77	2202	3793	94	2193	5052	78	2193	4513

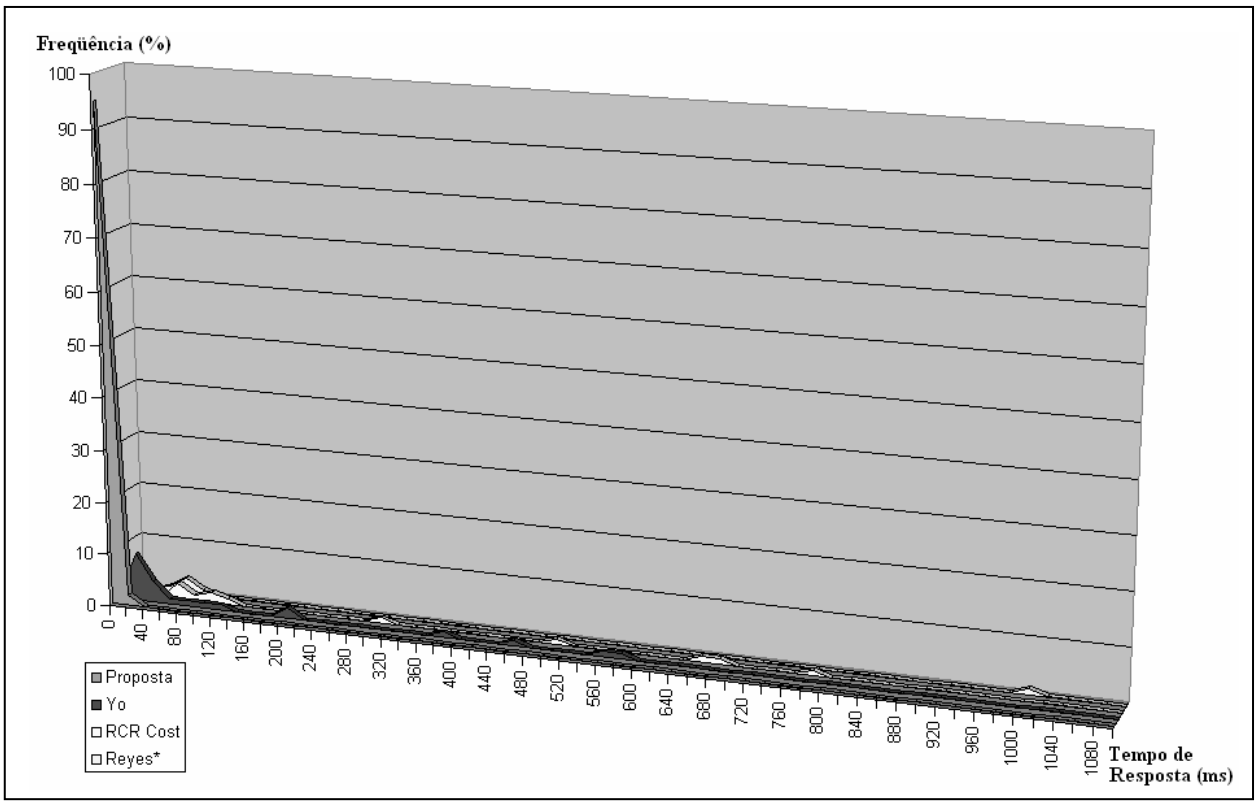

FIGURA 7.9 Curvas de freqüências dos tempos de respostas de sistema a partir das diferentes heurísticas.

Os números em termos absolutos são de pouca relevância para o estudo em questão, visto que a magnitude do tempo de resposta supõe-se depender do tamanho do problema e do tipo de busca usada. O tamanho do problema, em termos de produtos a serem fabricados, foi propositalmente reduzido para que um número considerável de experimentos pudesse ser obtido em tempo viável usando a busca A*.

O ganho relativo de desempenho neste parâmetro pode ser deduzido a partir dos dados de dispersão e medida central apresentados na tabela 7.2. Comparando os valores, observa-se uma média considerável na redução relativa do tempo de resposta, quase que na ordem de mil vezes o desempenho das heurísticas comparadas.

TABELA 7.2 Valores de medida central e de dispersão do tempo de resposta.

	Tempo de Resposta Obtido			
Heurística	***Média***	***Desvio Padrão***	***Mín.***	***Máximo***
Proposta	14,26ms	87,71ms	0,00ms	874,00ms
Yu	12.430,59ms	37.019,68ms	0,00ms	206.858,00ms
RCR Cost	15.055,64ms	37.354,56ms	46,00ms	239.000,00ms
Reyes*	15.060,89ms	35.824,96ms	46,00ms	204.031,00ms

Nos casos em que a heurística proposta demonstrou melhor tempo de resposta, evidenciou-se um número reduzido de nós de estado de busca explorados para se chegar a uma resposta. É senso comum que o tempo de resposta esteja relacionado com o número de nós explorados no processo de busca, mas esta relação não é linear. Esse "alívio" obtido no processo de busca diminui significativamente o *overhead* causado pela geração de nós e pelas comparações para a escolha do nó de custo mínimo estimado, aumentando a taxa média de nós explorados por segundo.

Observou-se que o *makespan* obtido pela heurística proposta não se distanciou dos encontrados nas demais heurísticas testadas. Igualou-se ao mínimo dentre elas em 24% dos casos. Sua distância máxima em relação ao menor valor encontrado, não foi maior que 10,13%, se distanciando em média de 1,48% do mínimo *makespan* encontrado nos testes. Em 91% dos casos, essa distância é menor que 3,5%. Preserva-se assim, até certo grau, o nível de optimalidade (proximidade da solução ótima) obtido em trabalhos anteriores.

CAPÍTULO 8 *Conclusão*

ALGUMAS heurísticas têm sido criadas para a redução do esforço na exploração de nós em uma busca sobre um modelo em rede de Petri na tentativa de se obter uma programação otimizada para um FMS. Mesmo quando as simplificações do modelo permitem definir uma condição ótima a ser atingida, o tempo de resposta na utilização de uma busca sobre os possíveis estados do sistema é inviável. Na tentativa de se obter, em um menor tempo, resultados de programação de FMS com considerável proximidade do grau de otimização que se observa em funções heurísticas de trabalhos anteriores, propõe-se uma nova heurística.

Primeiramente, foi dada uma definição formal do problema e foram definidas as etapas de modelagem correspondente usando a teoria de grafos e rede de Petri virtual. Aplica-se um busca sobre o grafo de alcançabilidade do modelo em redes de Petri usando o conceito de marcações temporizadas. Esta busca é aplicada para encontrar a seqüência de disparo de transições que corresponda às operações no chão de fábrica ao longo do tempo visando obter o menor tempo possível de produção.

Um sistema foi implementado, usando-se o algoritmo A* adaptado para seu uso sobre o grafo de alcançabilidade em rede de Petri. Foram implementadas quatro funções heurísticas para seu uso no busca: a heurística apresentada neste trabalho e outras três, correspondendo às propostas por Reyes *et al*, 2002, e Yu *et al*., 2003, e uma variante adaptada dessas duas últimas.

Os experimentos realizados indicaram, comparativamente, uma redução do número de nós explorados e, desta forma, um considerável ganho no tempo de resposta. O nível otimização se aproximou do obtido em trabalhos anteriores, chegando a ser a melhor em quase um quarto dos casos.

O uso conjugado de esforços na execução de um algoritmo de busca eficiente e de uma função heurística que permita reduzir o tempo de resposta pode propiciar a obtenção da programação da produção de um FMS em um curto espaço de tempo.

APÊNDICE A – Heurística de Reyes e variantes

A heurística de Reyes et al., 2003, é baseada em uma matriz com valores referentes aos custos mínimos de tempo que um produto em processo de fabricação pode vir a ter ao migrar de determinado *buffer* do sistema a outro – a chamada matriz de Alcançabilidade de Custo de Recurso (*Resource Cost Rechability – RCR*).

Trata-se de uma matriz quadrada cujas linhas e colunas referem-se aos lugares que representam *buffers* na rede de Petri obtida do sistema. Quando a "migração" de materiais entre dois *buffers* quaisquer do sistema não faz sentido no modelo do FMS, a respectiva posição na matriz *RCR* possui um valor não definido e, por isso, representado pelo símbolo ∞.

Seja Q o conjunto dos lugares que representam *buffers* e Rt o número de recursos de produção do sistema. Define-se *conjunto de lugares disponíveis* $AVA(M)$ como sendo o conjunto dos lugares $p \in Q$ de M, tais que $M(p) > 0$. Também, define-se $PAIRS(M,M')$ como sendo o conjunto dos pares de lugares (q, p) em que $q \in AVA(M')$ e $p \in AVA(M)$ tal que $\sum RCR(q, p)\ \forall (q, p) \in PAIRS\ (M,M')$ seja mínimo, se todos os lugares disponíveis em M' forem observados em $PAIRS(M,M')$. Assim, a heurística de Reyes é definida como:

$$h_{RCR}(M) = \frac{\sum RCR(p_{i'}, p_i)}{Rt}, \text{ tal que } (p_{i'},p_i) \in PAIRS(M, M_F)$$

Em Yu *et al.*, 2003, essa heurística é usada em conjunto com uma função de custo diferente da adotada na presente abordagem. Considerando os *tokens* $u = (p_u, \tau(u))$ da marcação temporizada, usou-se uma função de custo $j(n) = g(n) + \frac{\sum_i \tau(u_i)}{|U|}$, ou seja, adicionou-se *o tempo de retenção remanescente médio* $\overline{\tau_r}(\boldsymbol{M})$ dos *tokens* indisponíveis da marcação temporizada $\boldsymbol{M}$ para tratar o respectivo custo. Desta forma, a função de avaliação dos nós, o custo total estimado, fica sendo

$$f(n) = \left(g(n) + \overline{\tau_r}(n)\right) + h_{RCR}(n) = g(n) + \left(\overline{\tau_r}(n) + h_{RCR}(n)\right).$$

Cabe notar que os nós *n* de busca correspondem a marcações temporizadas ***M***. Assim, quando houver o termo *n* em uma fórmula qualquer, pode-se entender ***M*** e vice-versa.

Pode-se, assim, usar a heurística $h_{yu}(n)= \overline{\tau_r}(n) + h_{RCR}(n)$ na busca da forma como foi implementada no presente trabalho para a obtenção do mesmo comportamento em relação à exploração de nós usando a função de custo $j(n)$.

Uma variante da heurística de Reyes que se propõe para comparações é obtida substituindo-se $\overline{\tau_r}(n)$ de $h_{YU}(n)$ pelo *tempo de retenção remanescente máximo* $\tau_{MAX}(n)$ dentre os $\tau(u)$ dos *tokens* da marcação temporizada ***M***, sobre a qual a função heurística é aplicada. Assim, tem-se uma variante da heurística de Reyes, $h_{REYES*}(n) = \left(h_{RCR}(n) + \tau_{MAX}(n)\right)$.

Em suma, uma única heurística, $h_{RCR}(n)$, foi implementada para confrontar a heurística proposta, mas algumas variações são consideradas a partir de três possibilidades para $j(n)$:

$$j_1(n) = g(n) + \overline{\tau_r}(n),$$

$$j_2(n) = g(n) + \tau_{MAX}(n),$$

$$j_3(n) = g(n).$$

Originalmente a heurística de Reyes foi proposta para ser usada em buscas especialmente desenvolvidas para um melhor desempenho e, que embora eventualmente possam não encontrar a melhor solução, seus resultados aproximam-se do mínimo *makespan*.

APÊNDICE B – Resultados (Lista Completa)

	Proposta			Yu			RCR Cost			Reyes*		
#	T.Resp.(ms)	Makespam	Nr. Nós	T.Resp.(ms)	Makespam	Nr. Nós	T.Resp.(ms)	Makespam	Nr. Nós	T.Resp.(ms)	Makespam	Nr. Nós
1	0	1785	62	63	1765	3653	47	1765	3896	46	1765	4059
2	0	2262	48	2218	2262	24263	2781	2262	26481	1702	2262	22183
3	0	1820	138	139	1820	6633	204	1820	8810	203	1820	8219
4	0	2354	107	78	2360	4626	343	2360	11631	391	2354	12068
5	0	2870	174	77313	2839	98185	51593	2839	79395	71968	2839	93796
6	0	1826	341	32	1802	1955	186	1802	7870	171	1802	7460
7	0	1799	76	78	1731	3863	437	1731	12174	343	1731	10897
8	0	2817	174	157	2805	6202	592	2805	14486	579	2805	14307
9	0	2780	143	8719	2715	36955	10265	2703	39822	11265	2703	40904
10	0	1845	125	219	1883	8607	156	1883	6977	140	1883	6856
11	0	2206	549	124	2272	5911	1609	2181	20681	1703	2206	20926
12	15	3265	950	200108	3224	159555	239000	3224	170037	204031	3224	155595
13	15	2383	240	31	2269	2584	140	2269	6185	203	2269	8061
14	16	2715	679	3889	2715	27262	5875	2715	32278	5344	2715	31240
15	15	2331	466	281	2238	9468	531	2238	13177	391	2238	11296
16	874	3077	16207	63125	2794	90203	57797	2794	86916	47468	2794	80327
17	0	1836	203	920	1804	15960	2079	1804	21885	1764	1804	20321
18	15	2708	968	63375	2679	86052	81969	2679	99997	57609	2679	82773
19	15	2756	1005	1889	2710	20595	17610	2710	48864	18405	2684	50144
20	15	1841	85	0	1812	796	46	1812	2691	46	1807	2668
21	0	1835	131	63	1876	3114	264	1876	9616	249	1835	9129
22	0	2384	91	124	2384	5755	108	2384	5419	125	2384	5821
23	0	2223	477	312	2223	9991	5281	2247	31341	4483	2223	29338
24	15	2321	1687	15610	2237	46581	18092	2237	49476	18312	2237	44069
25	0	2413	64	93	2406	4648	249	2373	9155	218	2373	8834
26	0	2378	209	249	2316	9144	311	2316	10396	374	2339	11531
27	0	2413	132	297	2407	9755	186	2407	8154	219	2407	8425
28	0	2322	148	77	2202	3793	94	2193	5052	78	2193	4513
29	0	1909	136	31	1874	2322	219	1874	9140	217	1874	8608
30	0	2368	69	32	2310	1859	92	2292	4711	78	2310	4250
31	0	2745	121	14907	2713	46644	20546	2726	53338	19750	2689	52453
32	45	2265	2817	1640	2204	19399	20781	2204	56735	31983	2204	64145
33	0	2833	487	2907	2863	25515	2264	2863	23179	2297	2863	23186
34	15	2741	464	1250	2669	17761	17015	2669	49128	13499	2669	43784
35	16	2901	358	172296	2855	146187	141093	2855	128838	159234	2855	139287
36	0	2807	233	1764	2745	19871	14265	2726	45579	16281	2726	48011
37	15	2292	557	485	2294	12020	3781	2248	26838	3468	2248	26041
38	0	2824	131	124	2749	5396	234	2749	8842	297	2749	10065
39	0	1823	261	78	1823	4175	1093	1823	17486	952	1823	16687
40	0	1831	121	94	1782	4363	93	1782	5146	109	1799	5232
41	0	1862	258	702	1820	14982	780	1820	15598	546	1820	13844
42	16	2843	583	266	2800	9358	3108	2800	25986	2593	2800	24287
43	0	2362	159	2828	2354	24723	1874	2364	21563	2687	2337	24226
44	0	2849	155	1828	2775	21490	13578	2775	43544	15608	2775	46491
45	0	2343	128	17608	2328	51619	15656	2328	49546	15828	2328	50063
46	15	2185	1329	2202	2185	22310	3954	2185	27837	3406	2185	26121
47	15	2379	142	171	2342	7202	312	2342	10572	359	2342	11267
48	16	2690	435	57139	2690	86089	56422	2690	84012	54657	2690	82809
49	0	3196	467	1765	3200	20178	12734	3233	43915	8905	3140	37655
50	0	2267	234	329	2260	10141	3233	2260	25967	2983	2260	25357
51	16	2267	251	61	2248	3605	407	2243	12092	280	2243	10083
52	94	2661	4661	10171	2623	38711	80062	2623	104047	77172	2623	97354
53	0	2288	43	19046	2288	52624	21046	2288	55532	19797	2288	54629
54	0	2665	415	4389	2757	27526	29672	2665	66115	25266	2665	60299
55	30	3609	2356	206858	3498	157871	199780	3498	150763	190610	3498	146279
56	0	2742	237	1703	2771	19330	9187	2748	38963	9891	2748	39592
57	0	2789	148	531	2770	12466	594	2800	13397	577	2770	13306
58	15	2230	1079	9515	2230	38210	15938	2230	47642	12796	2230	43941
59	0	1924	813	391	1905	10942	297	1924	9762	328	1924	10659
60	0	1801	79	31	1801	2453	46	1801	2769	46	1822	2721
61	0	2383	170	2390	2353	23296	1718	2344	20647	1984	2344	22103
62	0	1943	149	499	1943	12586	485	1943	12718	499	1943	13169
63	15	2312	71	30	2358	2574	79	2300	4399	78	2300	4135
64	0	2847	224	421	2819	12255	7343	2819	36293	9686	2819	40705
65	0	2292	87	313	2298	10146	3327	2283	27452	3141	2283	26476
66	0	2710	174	61	2720	3316	406	2725	12206	358	2686	11320
67	0	2785	105	10453	2753	40323	4500	2753	29069	8514	2753	37412
68	0	1878	434	47	1818	2773	343	1818	11383	390	1805	12042
69	0	2276	151	937	2276	17388	1343	2278	19987	1187	2276	19009
70	16	2750	872	78436	2662	99872	99250	2660	110190	87359	2660	102005
71	0	2228	72	47	2232	2700	218	2232	8388	218	2204	8571
72	0	2840	96	1483	2834	19570	2422	2834	23313	1609	2834	20392
73	0	2810	379	1546	2816	18899	12891	2816	44622	14093	2795	46058
74	0	2252	314	46	2264	2690	234	2264	8852	219	2264	8618
75	0	1855	91	45	1868	2871	204	1868	8160	233	1868	8805
76	0	2713	187	78	2707	3935	422	2707	11948	265	2707	9468
77	0	1834	120	265	1834	9881	390	1834	11921	312	1834	10853
78	0	2359	50	249	2292	8863	1438	2292	20901	1608	2292	21928
79	0	2349	270	297	2326	10514	202	2326	8394	235	2326	9346
80	0	1868	154	30	1868	1452	46	1816	3585	62	1816	4035
81	0	2799	90	188	2779	7941	1561	2779	20918	702	2779	16012
82	0	2830	58	2234	2828	24408	9563	2792	41680	10000	2792	42349
83	0	1922	106	30	1878	2161	63	1878	3580	46	1900	3550
84	15	2225	286	516	2233	13265	10515	2146	42179	10062	2178	40882
85	0	2247	490	311	2206	9675	4313	2206	28433	4936	2206	29853
86	0	2293	105	719	2282	13329	1062	2282	15641	578	2282	12218
87	15	2305	1494	2593	2303	22192	1968	2303	20053	1797	2303	19872
88	0	1781	154	46	1853	3279	297	1853	9816	233	1776	9077
89	31	2195	1156	234	2191	8917	3514	2191	27120	4110	2191	29330
90	0	1828	113	31	1752	2202	140	1752	6901	187	1752	8440
91	15	1822	122	46	1841	3581	312	1837	11291	312	1837	11510
92	0	2277	129	311	2231	11007	328	2302	11294	297	2231	10642
93	0	1885	96	171	1863	7649	296	1863	11190	249	1876	9734
94	0	2846	423	33311	2759	70659	40250	2759	76155	47249	2759	80871
95	16	2757	892	110094	2756	115454	66234	2756	87613	101843	2756	111105
96	0	1757	254	188	1721	7669	3827	1721	26998	2750	1721	23846
97	0	3139	448	2733	3139	23786	23124	3139	55123	29703	3139	63123
98	0	2790	240	18656	2754	49997	12546	2771	42552	18265	2771	50476
99	0	1830	155	46	1800	2574	233	1800	9453	219	1784	8733
100	0	2394	128	203	2380	8106	156	2380	6522	171	2380	7200

Referências Bibliográficas

ABDALLAH, I. B.; ELMARAGHY, H.; ELMEKKAWY, T. Efficient search algorithm for deadlock-free scheduling in FMS using Petri nets. In: ROBOTICS AND AUTOMATION, 2., 1998, Leuven. **Proceedings of IEEE International Conference on.** San Diego: IEEE Computer Society Press, 1998. p. 1793-1798.

AGERWALA, T.; FLYNN, M. Comments on capabilities, limitations and "correctness" of Petri nets. In: COMPUTER ARCHITECTURE, 1., 1973, New York. **Proceedings of the 1st annual symposium on**. New York: ACM Press, 1973. p. 81-86.

BEASLEY, D.; BULL, D. R.; MARTIN, R. R. An Overview of Genetic Algorithms: Part 1, Fundamentals. **University Computing**, v. 15, n. 2, 58-69, 1993.

BOWDEN, F.D. A Brief Survey and synthesis of the Roles of Time in Petri Nets. **Mathematical and Computer Modeling**, v. 31, p. 55-68, 2000.

BUSI, N. Analysis issues in Petri nets with inhibitor arcs. **Theoretical Computer Science,** v. 275, n. 1-2, p.127-177, 2002.

CARVALHO, V. O. **Um Modelo de Seqüenciamento da Produção para um Sistema de Apoio à Decisão.** 2003. Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação, Universidade Federal de São Carlos, São Carlos.

CHEN, Y. L.; SUN, T. H.; FU, L. C. A Petri-Net Based Hierarchical Structure for Dynamic Scheduler of an FMS - Rescheduling and Deadlock Avoidance. In: ROBOTICS AND AUTOMATION, 3., 1994, San Diego. **Proceedings of IEEE International Conference on.** New York: IEEE Computer Society Press, 1994. p. 1998-2005.

CHEN, S. C.; JENG M. D. FMS Scheduling Using Backtracking-Free Heuristic Search Based on Petri Net State Equations. In: SYSTEMS, MAN, AND CYBERNETICS, 1995, Vancouver. **Proceedings of IEEE International Conference on**. 1995. p. 2153-2158.

CHEN, S. C.; JENG M. D. A Heuristic Approach Based on the State Equations of Petri Nets for FMS Scheduling. In: INDUSTRIAL AUTOMATION AND CONTROL, 1995, Taipei. **Proceedings of IEEE-IAS International Conference on**. Taipei, 1995, p. 275-281.

CHENG, C. W.; SUN, T. H.; FU, L.C. Petri-Net Based Modeling and Scheduling of a Flexible Manufacturing System. In: ROBOTICS AND AUTOMATION, 1., 1994, San Diego. **Proceedings of the 1994 International Conference on**. San Diego: IEEE Computer Society Press, 1994. p. 513-518.

CHIU Y.F.; FU, F.C. A GA embedded dynamic search algorithm over a petri net model for an fms scheduling. In: ROBOTICS AND AUTOMATION, 1., 1997, Albuquerque. **Proceedings of IEEE International Conference on.** New York: IEEE Computer Society Press, 1997. p. 513-518.

CONWAY, R. W.; MAXWELL, W. L.; MILLER, L. W. **Theory of Scheduling.** Reading, Massachusetts: Addison-Wesley, 1967. 294 p.

EICHENAUER, B. (1996) **A Procedure for a Unified Approach to Analysis and Development Projects**. Disponível em: IBE Simulation Engineering GmbH <http://www.ibepace.com>. Acesso em: 28 Jan. 2004.

ESSER, R. An Object Oriented Petri Net Language for Embedded System Design. In: INCORPORATING CASE'97, 1997, London. **8th International Worshop**. Anais. London, 1997. p. 216-223.

GAITHER, N.; FRAZIER, G. **Administração da produção e operações**. 8ª ed. São Paulo: Thomson Learning Inc., 2001. 598 p.

GANG, X.; WU, Z. A Kind of Deadlock-free Scheduling Method Based on Petri Net. In: HIGH ASSURANCE SYSTEMS ENGINEERING (HASE'02), 7., 2002, Tokyo. **Proceedings of 7th IEEE International Symposium on**. 2002. p. 195-200.

GANG, X.; WU, Z. Deadlock-free scheduling method using Petri net model analysis and GA search. In: CONTROL APPLICATIONS, 2., 2002, Glasgow. **Proceedings of the 2002 International Conference on.** IEEE Press, 2002. p. 1153-1158.

GANG, X.; WU, Z. Deadlock-Free Scheduling Strategy for Automated Production Cell. In: SYSTEMS, MAN, AND CYBERNETICS - PARTE A: SYSTEMS AND HUMANS, 34(1), 2004, Shanghai. **IEEE Transactions on.** Shanghai, 2004. p. 113-122.

GROOVER, M. P. **Automation, Production Systems, and Computer - Integrated Manufacturing**. Englewood Cliffs, NJ: Prentice-Hall, 1987.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence (Complex Adaptive Systems S.)**. Cambridge, MA, USA: MIT Press, 1975.

HOPCROFT, J.; ULLMAN, J. D. **Introduction to automata theory, languages, and computation.** Massachusetts: Addison-Wesley Publishing Company, 1939. 418 p.

JENG, M. D.; CHEN, S. C. Heuristic search approach using approximate solutions to Petri net state equations for scheduling flexible manufacturing systems. **International Journal of Flexible Manufacturing Systems**, v.10, n. 2, p. 139-162, 1998.

JENG, M. D.; CHEN, S. C. Heuristic search based on Petri net structures for FMS scheduling. In: INDUSTRY APPLICATIONS, 35(1), 1999. **IEEE Transactions on.** 1999. p. 196-202.

JENG, M. D.; LIN, C. S. Petri nets for the formulation of aperiodic scheduling problems in FMSs. In: EMERGING TECHNOLOGIES AND FACTORY AUTOMATION

PROCEEDINGS, 1997, Los Angeles. **6th International Conference on.** 1997. p. 375-380.

JENG, M. D.; JAW, R. W.; HUNG, P. L. Scheduling FMS with due dates based on Petri net state equations**.** In: SYSTEMS, MAN, AND CYBERNETICS, 1997, Orlando. **Proceedings of the 1997 IEEE International Conference on.** Anais. Orlando, 1997. p. 2724-2729.

LEE, D. Y.; DICESARE, F. FMS scheduling using Petri nets and heuristic search**.** In: ROBOTICS AND AUTOMATION, 1992, Nice. **Proceedings of the 1992 IEEE International Conference on.** Anais. Nice, 1992. p. 1057-1062.

LEE, D. Y.; DICESARE, F. Integrated models for scheduling flexible manufacturing systems. In: ROBOTICS AND AUTOMATION, 1993a, Atlanta. **Proceedings of the IEEE International Conference on**. Anais. Atlanta, 1993. p. 827-832.

LEE, D. Y.; DICESARE, F. Scheduling flexible manufacturing systems with the consideration of setup times. In: DECISION AND CONTROL, 1993b, San Antonio. **Proceedings of the 32nd IEEE Conference on.** Anais. San Antonio, 1993. p. 3264-3269.

LEE, D. Y.; DICESARE, F. Scheduling flexible manufacturing systems using Petri nets and heuristic search. **IEEE Transactions on Robotics and Automation**, v.10, n. 2, 1994. p. 123-133.

LIN, T. Y. Extensions of pushdown automata and Petri nets. In: ACM ANNUAL COMPUTER SCIENCE CONFERENCE, 1989, Louisville. **Proceedings of the 17th conference on.** ACM, 1989. p. 414-414.

LIN, C. et al. A Sufficient Condition for Instability of Buffer Priority Policies in Re-Entrant Lines. **IEEE Transactions on Automatic Control**, v.48, n. 7, p. 1235-1238, 2003.

LIN, C.; XU, M.; MARINESCU, D. C. A Petri net approach to stability analysis of buffer priority scheduling policies in manufacturing systems. In: SYSTEMS, MAN AND CYBERNETICS, 2001, Tucson. **IEEE International Conference on**. Anais. Tucson, 2001. p. 1811-1816.

MORANDIN JÚNIOR, O. **Metodologia de Modelagem de Sistemas Flexíveis de Manufatura, utilizando Rede de Petri Virtual**. 1999. 139 f. Tese (Doutorado em Engenharia Mecânica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

MORANDIN JÚNIOR, O.; KATO, E. R. R. Virtual Petri nets as a modular modeling method for planning and control tasks of FMS. In: SYSTEMS, MAN, AND CYBERNETICS, 2003, Washington. **Proceedings of the 2003 IEEE International Conference on**. Anais. Washington, 2003. p. 1521-1527.

MORTENSEN, K. H.; RÖLKE, H. (2001) A Classification of Petri Nets. Disponível em: Petri Net World <http://www.daimi.au.dk/PetriNets/classification/>. Acesso em: 14 Fev 2004.

MURATA, T. Petri Nets: Properties, Analysis and Applications. **Proceedings of the IEEE**, v.77, n. 4, p. 541-580, 1989.

PEARL, J. **Heuristics: Intelligent Search Strategies for Computer Problem Solving**. Massachusetts: Addison-Wesley, 1981. 382 p.

PETERSON, J. L. Petri Nets**. Computing Surveys**, v.9, n. 3 p. 223-252, 1977

PETERSON, J. L. **Petri Net Theory and the Modeling of Systems.** Prentice Hall, Inc, 1981, 288 p.

REYES-MORO, A. YU, H.; KELLEHER; LOYD, S. Integrating Petri nets and hybrid heuristic search for the scheduling of FMS. **Computers in Industry**, v.47, n. 1, 2002. p. 123-138.

REYES-MORO, A.; YU, H.; KELLEHER, G. Advanced scheduling methodologies for flexible manufacturing systems using Petri nets and heuristic search. In: ROBOTICS AND AUTOMATION, 2000, San Francisco. **IEEE International Conference on.** Anais. San Francisco, 2000. p. 2398-2403.

REYES-MORO, A.; YU, H; LOYD, S. An Evolutionary Hybrid Scheduler based in Petri Net Structures for FMS Scheduling. In: SYSTEMS, MAN, AND CYBERNETICS, 2001, Tucson. **Proceedings of IEEE International Conference on**. NY: IEEE, 2001. p. 2516-2521.

REYES-MORO, A., YU, H.; KELLEHER, G. Hybrid heuristic search for the scheduling of flexible manufacturing systems using Petri nets. In: ROBOTICS AND AUTOMATION, 18(2), 2002. **IEEE Transactions on.** 2002. p. 240-245.

RUSSEL, S. & NORVIG, P. **Inteligência Artificial: Um enfoque moderno**. Tradução da 2ª. ed. São Paulo: Prentice Hall. 2004. 1021 p.

SLACK, N. et al. **Administração da Produção**. São Paulo: Editora Atlas S.A, 1999. 526 p.

TEMPELMEIER, H.; KUHN, H. **Flexible Manufacturing Systems - Decision Support for Design and Operation**. New York: John Wiley & Sons, Inc, 1993. 488 p.

TSUKAMOTO, Y. An Approach to Fuzzy Reasoning Method. In: GUPATA, M. M.; RAGADE, R. K.; YAGER, R.R. **Advances in Fuzzy Set Theory and Applications.** Amsterdam: North-Holland, 1979.

TUBINO, D. F. **Manual de Planejamento e Controle da Produção**. São Paulo: Editora Atlas. 2000.

VALETTE, R.; PRANDIN-CHÉZALVIEL, B.; GIRAUT, F. An Introduction to Petri Net Theory. In: CARDOSO, J.; CAMARGO, H. **Physica-Verlag Fuzziness in Petri Nets - Studies in Fuzziness and Soft Computing**. Heidelberg: Physica-Verl., 1999. p. 3-24.

XIONG, H. H.; ZHOU, M.; MANIKOPOULOS, C. N. Scheduling flexible manufacturing systems based on timed petri nets and fuzzy dispatching rules. In: EMERGING TECHNOLOGIES AND FACTORY AUTOMATION, 3., 1995, Paris. **Proceedings of the 1995 INRIA/IEEE Symposium on**. Paris, 1995. p. 309-315.

XU, J. X. Fuzzy Petri net-based optimum scheduling of FMS with human factors. In: DECISION AND CONTROL, 1996, Kobe. **Proceedings of the 35th IEEE Conference on**. Anais. Kobe, 1996. p. 4451-4452.

YIM, S. J.; LEE, D. Y. Multiple objective scheduling for flexible manufacturing systems using Petri nets and heuristic search. In: SYSTEMS, MAN AND CYBERNETICS, 4., 1996, Beijing. **Proceedings of the 1996 IEEE International Conference on.** Anais. Beijing, 1996. p. 2984-2989.

YU, H. et al. Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems - Part I. Petri net modelling and heuristic search. **Computers and Industrial Engineering**, v. 44, n. 4, p. 527-543, 2003a.

YU, H. et al. Combined Petri net modelling and AI-based heuristic hybrid search for flexible manufacturing systems - Part II. Heuristic hybrid search. **Computers and Industrial Engineering**, v. 44, n. 4, p. 545-566, 2003b.