DENISE SODERO VINHAS PORTUGAL

# Modelagem e Programação de Sistemas a Eventos Discretos Periódicos

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos pré-requisitos para obtenção do Título de Doutor em Engenharia Elétrica.
Área de concentração: Automação Industrial.

*Orientador:* PROF. DR. RAFAEL SANTOS MENDES

**Campinas**
**2006**

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Portugal , Denise Sodero Vinhas
P838m Modelagem e programação de sistemas a eventos discretos periódicos / Denise Sodero Vinhas Portugal. -- Campinas, SP: [s.n.], 2006.

Orientador: Rafael Santos Mendes.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes de Petri. 2. Processos de fabricação - Automação. 3. Escalonamento de produção. 4. Otimização combinatória I. Mendes, Rafael Santos. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Titulo em Inglês: Modelling and programming of periodic discrete events systems.
Palavras-chave em Inglês: Petri net, Industrial automation, Cyclic scheduling, Combinatorial optimization
Área de concentração: Automação industrial.
Titulação: Doutora em Engenharia Elétrica
Banca examinadora: Jean-Marie Farine, Carlos Alberto dos Santos Passos, Akebo Yamakami, Márcio Luiz de Andrade Netto, Fernando Antonio Campos Gomide.
Data da defesa: 30/10/2006

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

# Modelagem e Programação de Sistemas a Eventos Discretos Periódicos

Este exemplar corresponde à redação final
da tese devidamente corrigida e defendida
por **Denise Sodero Vinhas Portugal** e aprovada
pela comissão julgadora em 30 de outubro de 2006.

**Prof. Dr. Rafael Santos Mendes**
**Orientador**

**Prof. Dr. Jean-Marie Farine**

**Dr. Carlos Alberto dos Santos Passos**

**Prof. Dr. Akebo Yamakami**

**Prof. Dr. Márcio Luiz de Andrade Netto**

**Prof. Dr. Fernando Antonio Campos Gomide**

## Resumo

Uma metodologia para obter um escalonamento cíclico em Sistemas a Eventos Discretos é proposta neste trabalho. Esta metodologia parte de uma rede de Petri que modela minimamente um sistema a eventos discretos funcionando em regime periódico. O método identifica quais são as redes que podem ser tratadas por ele. As redes de Petri *tratáveis* serão decompostas em subredes identificadas por processos, que são classificados de acordo com suas topologias, o que permite a modelagem do escalonamento cíclico do sistema através de uma modelagem em programação linear inteira mista. Este modelo em MILP será implementado no *software* GAMS. Alguns exemplos tirados da literatura serão usados para mostrar e testar a aplicação desta metodologia.



## Abstract

A methodology to obtain a cyclic scheduling in Discrete Events Systems is proposed in this work. This methodology initializes with a Petri net modeling a discrete events system functioning with periodic processing. The method identifies which are the nets that can be treaties by him. The "tractable" Petri nets will be decomposed in subnets identified by process, which are classified according to its topologies, that permits us to model the cyclic scheduling of the system by a mixed integer linear programming model. This model in MILP will be implemented using software GAMS. Some examples from the literature will be used to show and to test the application of this methodology.

*À minha amiga Alessandra Ambrósio*
*in memorian*

# Agradecimentos

À minha mãe M. Ester que com seu exemplo de fé na vida, de dedicação ao trabalho, de amor à familia e com sua infinita paciência diante das dificuldades que surgem, me inspirou a chegar até aqui;

Ao meu querido esposo Rodrigo, por seu apoio incondicional, por seu amor, amizade e carinho que me fortalecem a cada dia;

Às minhas irmãs, M. Cecília, Solange e Ebe, pela imensa amizade e companheirismo e, sobretudo, por me fazerem ultrapassar os meus próprios limites;

Ao professor Rafael, por sua confiança em mim, por sua paciência e dedicação, e por sua habilidade em me orientar;

À minha amiga Mábia, por seu apoio e por sua ajuda inestimável na preparação deste trabalho;

À Unicamp e à FEEC por sua estrutura e ambiente motivadores;

À Capes, pelo suporte financeiro;

Enfim, a todos aqueles que me ajudaram um pouquinho aqui ou ali para a realização desta tese

*Agradeço*

# Sumário

# Relação de Figuras

# Relação de Tabelas

# 1 Introdução

Nas últimas décadas, avanços na tecnologia de automação da manufatura e no gerenciamento da produção, ocasionaram grandes mudanças tanto nos equipamentos utilizados nas indústrias de manufatura quanto nos estudos relacionados aos problemas de controle e de planejamento da produção. Por esta razão e também por sua grande importância na indústria, os estudos de automação dos processos e de alocação de recursos vem se tornando cada vez mais importantes. Exemplos de sistemas que sofreram estas mudanças estão por toda a parte, como: sistemas de redes de computadores e de comunicação, sistemas de manufatura automatizados, sistemas de controle em fábricas automotivas, sistemas de controle de tráfego aéreo, etc..

Estes sistemas são compostos por tarefas que são governadas por regras operacionais muitas vezes definidas por pessoas, portanto a dinâmica destes sistemas se caracteriza pela ocorrência assíncrona de eventos discretos, de maneira instantânea, sendo que o estado de tais sistemas só muda devido à ocorrência de algum evento (CASSANDRAS & LAFORTUNE, 1999). Por esta razão tais sistemas recebem o nome de Sistemas a Eventos Discretos (SED), em oposição aos sistemas de variáveis contínuas, tratados pela Teoria de Controle clássica (CURY, 2001).

Segundo Santos-Mendes (1995), *a História da Engenharia sempre foi marcada pelo paradigma de sistemas contínuos. Estes modelos formais, oriundos do estudo da Física, baseiam-se em equações diferenciais ou a diferenças. Porém, os sistemas a dinâmica discreta resistem a qualquer modelagem por estas equações. A existência desses sistemas e sua relevância cada vez maior levaram ao desenvolvimento de técnicas de análise, mesmo na ausência de um quadro teórico satisfatório. Assim, tradicionalmente estes sistemas foram tratados através de heurísticas e simulação. Embora nesta área o conhecimento disponível até o momento seja privilegiadamente numérico, há um grande esforço da comunidade científica no sentido de se desenvolver modelos analíticos. Estes problemas adquiriram importância ainda maior, conduzindo a um grande esforço teórico para o desenvolvimento de ferramentas formais mais adequadas. Este estudo encontra-se na confluência de três áreas: a Pesquisa Operacional, a Teoria de Controle e a Teoria de Computação (HO, 1989). Como consequência desta pluraridade de origens e do pouco tempo de existência, tem-se um quadro atual bem distante de uma desejável situação de unificação. Observa-se um grande número de abordagens, algumas delas com ferramentais e linguagens bem diferenciados. Esta multiplicidade, embora dispersiva, denota uma área de pesquisa vigorosa que se propõe a responder questões intelectualmente excitantes e de grande importância prática.*

O estudo e a programação de sistemas a eventos discretos periódicos que serão realizados no decorrer deste trabalho se inserem neste contexto, com o desenvolvimento de ferramentas for-

mais adequadas para a modelagem destes sistemas. Isto será feito em dois níveis. No primeiro nível, visando estabelecer relações funcionais entre os diversos componentes do sistema e verificar as propriedades essenciais, serão utilizadas as Redes de Petri. No segundo nível, visando a otimização de um escalonamento cíclico será utilizada a programação linear inteira mista.

As redes de Petri são ferramentas gráficas e matemáticas, sendo mais utilizadas na análise de comportamento, na avaliação de desempenho e na análise e verificação formal do comportamento de sistemas discretos. Uma referência básica no estudo de redes de Petri é o trabalho de Peterson (1981) que apresenta a teoria de redes de Petri e a modelagem de sistemas. Já Murata (1989) apresenta um tutorial com as principais propriedades e técnicas de análise de redes de Petri, que inclui a uma breve história das redes de Petri. Balbo *et. al.* (2000) também apresentam um tutorial introdutório a respeito de redes de Petri, incluindo redes de Petri coloridas e redes de Petri estocásticas. Diversas formas de aplicação de redes de Petri em sistemas industriais podem ser vistas em DiCesare *et. al.* (1993), Desrochers & Al-Jaar (1995) e Zurawski & Zhou (1994), enquanto Zhou & DiCesare (1993) apresentam um método de síntese de redes de Petri para o controle de sistemas a eventos discretos em sistemas de manufatura, e Wang (1998) apresenta a teoria de redes de Petri temporizadas e suas aplicações.

Com a grande utilização e diversidade das redes de Petri, nesta última década, a comunidade científica se reuniu para propor a padronização das redes de Rede de Petri. A primeira parte que trata dos conceitos, das definições e das notações gráficas de redes de Petri de Alto Nível já está disponível através da ISO 15909-1 (2004). A segunda parte que trata dos formatos de transferência dos programas computacionais ainda está em discussão (ISO 15909-2, 2005).

As redes de Petri constituem uma das abordagens para o estudo dos sistemas a eventos discretos, assim como, a álgebra de dióides, as cadeias de Markov, a teoria de filas e a teoria de controle supervisório (CASSANDRAS & LAFORTUNE, 1999).

As principais vantagens da utilização de redes de Petri no estudo de sistemas dinâmicos a eventos discretos são:

- possibilidade de serem expressas graficamente;
- possibilidade de descrição da ordem parcial entre vários eventos, levando-se em conta a flexibilidade do sistema, capturando suas relações de precedência e os vínculos estruturais dos sistemas reais;
- modelagem de conflitos e filas, além da descrição precisa e formal das sincronizações;
- possibilidade de representação explícita dos estados e dos eventos;
- possibilidade de uso na especificação, na modelagem, na análise, na avaliação do desempenho e na implementação desses sistemas;
- possibilidade de uso nos diversos níveis de estrutura hierárquica do controle, facilitando a integração destes níveis;
- existência de fundamentação matemática e prática;
- existência de várias especializações (RdP temporizadas, coloridas, estocásticas, etc.).

Em problemas de escalonamento da produção (alocação de operações a cada máquina), as Redes de Petri adaptam-se muito bem quando se estudam regimes estacionários (regimes regulares) com alimentação periódica (CURY, 1990).

Não existe uma maneira única para modelar Sistemas a Eventos Discretos através de Redes de Petri, existindo vários métodos de síntese (ZHOU & DICESARE, 1993) e portanto diferentes representações para um mesmo sistema.

Outra limitação das redes de Petri é que elas não possuem recursos para tomar decisões sobre os conflitos existentes no sistema. As redes podem explicitar os conflitos, mas não decidi-los. Este trabalho cabe ao programador/modelador.

Uma das metas desta tese é superar esta limitação e para isso serão propostas ferramentas para ajudar na tomada de decisão.

As limitações observadas para as redes de Petri em relação às tomadas de decisão, podem ser tratadas através da combinação desta classe de modelos com técnicas de tomada de decisão.

Segundo Luna (1995), *decisões e ações são o produto final do trabalho de gerentes, executivos e engenheiros. Tomada de decisão é o ato de selecionar entre todas as possíveis alternativas, a melhor solução para alcançar um determinado objetivo. Precede esta escolha final um processo complexo e, muitas vezes, demorado de exploração e análise do problema e da situação. Este processo é conhecido como* **processo decisório ou processo de tomada de decisão***, sendo que a Pesquisa Operacional (PO) é a sua ferramenta mais tradicional.*

A Pesquisa Operacional proporciona uma abordagem científica para tomada de decisões, baseada essencialmente em técnicas quantitativas, ao invés de uma abordagem qualitativa, baseada na experiência e intuição de quem toma as decisões. Estes sistemas são descritos e analisados em termos numéricos, possuem restrições, como por exemplo, limitação de recursos, tem objetivos a otimizar e não possuem solução imediata.

O processo de tomada de decisão em pesquisa operacional consiste em construir um modelo de decisão e resolvê-lo de modo a encontrar-se a melhor decisão ou decisão ótima. Apesar da exatidão da solução, para que esta solução tenha significado real, esse modelo deve proporcionar uma representação adequada da realidade. Seus principais modelos são os de programação matemática: programação linear e não-linear, programação inteira, programação dinâmica, teoria de jogos; e os modelos probabilísticos baseados: na teoria da probabilidade, na teoria de filas e na teoria de estoque (HILLIER & LIEBERMAN, 1988).

A programação matemática geralmente lida com problemas de alocação de recursos e de transporte, que são problemas típicos de escalonamento. Dentro da classe de problemas de escalonamento encontra-se o Escalonamento Cíclico, e dentro dela os problemas de escalonamento em sistemas de produção em regime periódico.

É necessário salientar que existe uma diferença entre escalonamento cíclico e escalonamento periódico, dada pelo fato que todo escalonamento periódico é cíclico, mas um escalonamento cíclico pode estar associado a um ciclo que não se repete em intervalos constantes ou conhecidos, isto é, apenas se repete. Este é o caso do escalonamento cíclico de salas de cirurgias em um hospital (OOSTRUM, 2006), onde se busca escalonar as salas de cirurgias, mas como cada uma destas salas tem tempo de ocupação não determinístico, não é possível estabelecer um escalonamento periódico para elas, sendo possível apenas escalonamentos cíclicos. Os problemas de escalonamento cíclico que serão estudados neste trabalho serão os que também são escalonamentos periódicos, por isso, no que se segue, os escalonamentos cíclicos também serão considerados periódicos.

O Escalonamento Cíclico é usado para referenciar uma grande quantidade de problemas de escalonamento periódico em um ambiente de demanda contínua e constante, tanto na área de produção quanto na de computação e o que pode ser observado pelo gradativo aumento de estudos a este respeito é que com o aperfeiçoamento da automação industrial e da computação paralela, tem sido cada vez maior a busca de melhorias através do Escalonamento Cíclico. Isto pode ser visto nos trabalhos de Crama *et. al.* (2000) e de Gultekin *et. al.* (2006) que lidam com células robóticas, e no trabalho de Dawande *et. al.* (2005) que estudaram os problemas de sequenciamento e escalonamento em células robóticas, dando grande destaque à produção cíclica de tarefas (partes de um processo). Segundo Dawande *et. al.* (2005), *A principal motivação para o estudo da produção cíclica vem da prática: escalonamentos cíclicos são fáceis de implementar e controlar e são a forma mais simples de especificar operações em células robóticas na indústria*.

Dauscha *et al.* (1985), Serafini & Ukovich (1989) e Roundy (1992) são referências básicas para estudos nessa área, pois nestes artigos são definidas ferramentas teóricas relevantes para solucionar alguns tipos de problemas de escalonamento cíclico. Dauscha *et al.* (1985) estudou o problema de sequenciamento cíclico com uma aplicação para o controle de tráfego. Serafini & Ukovich (1989) propôs um modelo matemático para problemas de escalonamentos periódicos. Roundy (1992) estabeleceu uma caracterização da *estrutura de precedência cíclica* usada para desenvolver um procedimento de busca **Branch & Bound** especializado para identificar este tipo de estrutura.

Algumas modelagens em programação matemática de problemas de escalonamento cíclico foram propostas, como por exemplo, por Rao (1992) que estudou estes problemas em ambientes de manufatura e por Hanen (1994) que estudou o problema do *job shop* recorrente. Já Song & Lee (1998) usaram a modelagem em redes de Petri para solucionar um job shop periódico onde cada máquina tem um *buffer* de entrada com capacidade limitada, propondo também, uma modelagem em programação inteira mista para achar um escalonamento ótimo, livre de impasse (deadlock) e com ciclo mínimo.

Problemas clássicos da área de escalonamento, como o *job shop* e o *flow shop*, também foram estudados em um ambiente de demanda contínua e constante. Por exemplo, Kats & Levner (1997) estudaram um problema de escalonamento cíclico de jobs idênticos em um *flow shop* reentrante, para problemas de escalonamento de job shops periódicos, Song & Lee (1996) desenvolveram um procedimento de busca tabu; enquanto Lee & Posner (1997) escalonaram e fizeram medidas de performance de alguns desse problemas. Steiner & Xue (2006) estabeleceram uma conexão entre o problema do *job shop* cíclico e o problema do *flow shop* recorrente. O problema de *job shop cíclico* também tem sido muito pesquisado. Portugal (1999) estudou as modelagens de Rao (1992) e de Hanen (1994) e as aplicou ao problema de job shop cíclico com jobs distintos.

A complexidade do *job shop cíclico* foi provada ser NP-difícil (KAMOUN & SRISKANDARAJAH, 1993; MUNIER, 1996 e VERHAEGH *et al.*, 1998). Roundy (1992) estudou a estrutura combinatorial dos escalonamentos cíclicos e provou que sua complexidade é NP-difícil.

Devido a dificuldade em resolver os problemas de escalonamento cíclicos pelos métodos tradicionais, muitos algoritmos para a resolução de escalonamentos cíclicos também foram propostos. Este é o caso do algoritmo proposto por Brucker & Kampmeyer (2005) que propuseram algoritmos de busca tabu para problemas de escalonamento de máquinas cíclicas, sendo que estes algoritmos utilizaram a modelagem proposta por Hanen (1994). Este também é o caso de Hsu *et. al.* (2002) e

Cavory *et. al.* (2005) propuseram algoritmos genéticos para resolver problemas de escalonamento cíclico.

As redes de Petri também são usadas para auxiliar na resolução do problema de escalonamento cíclico de sistemas de manufatura repetitiva e em robótica, como pode ser visto nos trabalhos de Freedman (1991), Tuncel & Bayhan (2003) e Lee (2005), entre outros. Porém, a forma mais comum de utilização das redes de Petri em problemas de manufatura repetitiva, é o uso destas redes em conjunto com modelagens em programação matemática. Isto pode ser visto em Trouillet & Benasser (2002), Chauvet *et. al.* (2003), em Jerbi *et. al.* (2004) e em Nakamura *et. al.* (2006), entre outros.

É interessante notar que as redes de Petri são usadas para modelar vários tipos de sistemas a eventos discretos com funcionamento periódico, mas não existe um modelo geral que abrange estes diversos tipos de sistemas.

## 1.1 Objetivos da tese

Como dito na seção anterior, os sistemas a eventos discretos periódicos podem ser modelados por redes de Petri e os problemas de escalonamento cíclico podem ser modelados através da técnicas de pesquisa operacional.

Apoiada nestas observações, esta tese tem como objetivo principal escalonar periodicamente, com o menor período possível, sistemas a eventos discretos. Para isto será proposta uma metodologia de avaliação, de planejamento e de escalonamento destes sistemas, baseada em redes de Petri e em programação matemática.

Desta forma, as redes de Petri descreverão a ordem parcial entre os vários eventos, levando em conta a flexibilidade do sistema, capturando suas relações de precedência e seus vínculos estruturais, fornecendo uma visão de conjunto do sistema. Já a programação matemática, aproveitará a descrição do sistema feita pela rede de Petri para modelar o problema de escalonamento cíclico deste sistema, resolvendo os conflitos explicitados pela rede de Petri.

Assume-se neste trabalho que os sistemas tratados já estarão modelados através de redes de Petri (isto é, o problema de síntese em redes de Petri não será tratado neste estudo).

Um algoritmo de avaliação de redes de Petri será proposto para verificar se o sistema possui um comportamento periódico e se satisfaz algumas propriedades (tais como, conservação e consistência).

Depois da etapa da avaliação, as redes de Petri serão utilizadas na elaboração da modelagem do problema de escalonamento cíclico de sistemas a eventos discretos.

Finalmente, o problema de escalonamento cíclico será resolvido através de um método exato, utilizando-se o software GAMS (*Generalized Aplied Modelling Systems*) com o *solver* OSL, que é uma linguagem de alto nível muito utilizada por pesquisadores e pela indústria para a resolução de modelos de Pesquisa Operacional. Este programa resolve problemas de programação linear inteira mista através da técnica de *Branch & Bound*.

## 1.2 Estrutura do trabalho

A estruturação deste trabalho é mostrada a seguir.

O Cap. 2 apresentará os conceitos básicos essenciais para este trabalho.

O Cap. 3 se aprofundará em Redes de Petri, mostrando as diversas classes e subclasses, as principais propriedades e características de redes de Petri que, em geral, são usadas para modelar problemas em sistemas a eventos discretos. No final deste capítulo serão mostradas as propriedades que as redes de Petri devem satisfazer que tornam possível modelar o sistema associado à rede de Petri pelo modelo em programação inteira mostrado do Cap. 4.

O Cap. 4 tem como proposta estudar o comportamento periódico de sistemas a eventos discretos e propor uma nova modelagem matemática para problemas de escalonamento cíclico.

O Cap. 5 aborda as técnicas para a avaliação de Redes de Petri para Sistemas Periódicos e a para a modelagem destes sistemas em programação linear inteira mista (MILP).

Finalmente no Cap. 6 são apresentados alguns exemplos de utilização da metodologia proposta nos capítulos anteriores usando-se redes de Petri extraídas da literatura.

# 2

# Conceitos Básicos, Definições e Objetivos da Tese

Este capítulo visa apresentar os principais conceitos e definições de sistemas dinâmicos a eventos discretos que formam a base teórica utilizada nesta tese e que tornam possível a nossa definição de sistemas cíclicos.

Conforme discutido anteriormente, este trabalho visa o estudo de Sistemas Dinâmicos a Eventos Discretos com comportamento cíclico. Serão mostrados ainda aqui, os principais conceitos e definições de Redes de Petri, pois essas redes serão usadas neste trabalho como ferramentas de base para a modelagem dos Sistemas a Eventos Discretos. Como o material sobre Redes de Petri é extenso, o capítulo 3 completará o estudo sobre Redes de Petri mostrando um resumo sobre as classes, subclasses, principais características e propriedades de Redes de Petri, conceitos necessários ao desenvolvimento deste trabalho.

## 2.1 Sistemas a Eventos Discretos e Sistemas de Manufatura

### 2.1.1 Sistemas a Eventos Discretos

Os conceitos e definições mostrados nesta seção são baseados no trabalho de Santos-Mendes (1995) e de Cury (2001).

Os **Sistemas a Dinâmica Discreta** ou Sistemas a Eventos Discretos ou ainda Sistemas Discretos são caracterizados por três condições básicas: apresentam variáveis de estado discretas, são dirigidos a Eventos e não são descritos adequadamente por equações diferenciais (ou a diferenças) (CASSANDRAS & LAFORTUNE, 1999). A rigor, a primeira condição relacionada não caracteriza propriamente um sistema discreto, visto que sua inobservância não impede que um sistema apresente dinâmica discreta. O segundo ponto é talvez o mais importante da caracterização dos sistemas discretos. Sua dinâmica é dirigida a eventos, ou seja, o que determina a evolução do sistema é a ocorrência de eventos e não simplesmente o passar do tempo. São exemplos de eventos o início e o término de uma tarefa, a chegada de um cliente a uma fila ou a recepção de uma mensagem em um sistema de comunicação. A ocorrência de um evento causa, em geral, uma mudança interna no sistema. Além disso, uma mudança pode ser causada pela ocorrência de um evento interno ao próprio sistema, tal como o término de uma atividade ou o fim de uma temporização. Em qualquer caso, essas mudanças se caracterizam por serem abruptas e instantâneas:

ao perceber um evento, o sistema reage imediatamente, acomodando-se em tempo nulo em uma nova situação, onde permanece até que ocorra um novo evento. Desta forma, a simples passagem do tempo não é suficiente para garantir que o sistema evolua; para tanto, é necessário que ocorram eventos, sejam estes internos ou externos.

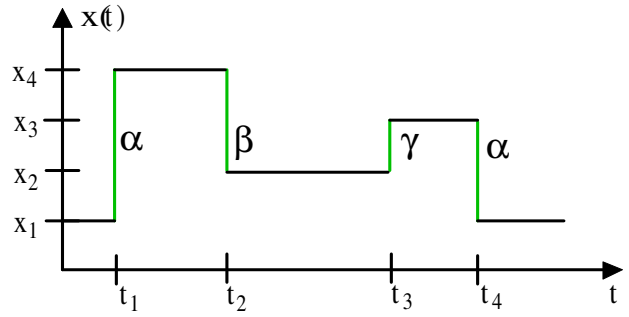



**Figura 2.1**: Trajetória típica de um sistema a eventos discretos (Cassandras & Lafortune, 1999).

Os sistemas tratados neste estudo são os sistemas dinâmicos a eventos discretos. Como descrito anteriormente, estes sistemas percebem as ocorrências no mundo externo através da recepção de estímulos, denominados eventos. A ocorrência de um evento causa uma transição ou mudança de estado no sistema e o sistema permanece neste estado até que algum outro evento ocorra. A evolução no tempo da ocorrência de eventos pode ser representada pela trajetória percorrida no seu espaço de estados, conforme ilustrado pela figura 2.1.

Nesta trajetória ocorrem eventos representado pelas letras $\alpha$, $\beta$ e $\gamma$. Note que um mesmo evento pode ter efeitos diferentes, dependendo do estado em que ocorre. Por exemplo, se o sistema está no estado $x_1$ e ocorre o evento $\alpha$, o próximo estado será $x_4$; se $\alpha$ ocorre em $x_3$, volta-se para $x_1$. A trajetória pode continuar indefinidamente, inclusive com a ocorrência de outros eventos, não representados na figura 2.1. Para todos os sistemas tratados, porém, assume-se que o número total de eventos diferentes que podem ocorrer é finito. Os sistemas a eventos discretos podem permanecer um tempo arbitrariamente longo em um mesmo estado, sendo que sua trajetória pode ser dada por uma lista de eventos na forma $\{\sigma_1, \sigma_2, \ldots\}$, incluindo-se eventualmente os instantes de tempo em que tais eventos ocorram, na forma $\{(\sigma_1, t_1), (\sigma_2, t_2), \ldots\}$. A quantidade de informação necessária depende dos objetivos da aplicação. O estado em que o sistema se encontra antes de ocorrer o primeiro evento recebe o nome de estado inicial. Em geral, é possível forçar um sistema a voltar a esse estado antes de iniciar sua utilização para um determinado fim. Este processo é denominado de reinicialização. Donde, pode-se concluir que especificar o comportamento de um sistema a eventos discretos é estabelecer seqüências ordenadas de eventos que levem à realização de determinados objetivos.

Em síntese, é possível relacionar algumas características que, de maneira essencial ou acessória, encontram-se presentes nos sistemas discretos de modo geral:

- estado discreto;
- sincronismo;
- concorrência;

- paralelismo;
- dinâmica dirigida a eventos.

É importante ressaltar que parte dos sistemas de interesse em Engenharia apresenta os elementos descritos e que estes não podem ser descritos pelos métodos tradicionais utilizando equações diferenciais.

Segundo Cury (2001), *vários modelos para SEDs foram desenvolvidos, sem que algum deles tivesse se firmado como universal. Esses modelos refletem tipos diferentes de SEDs inclusive com objetivos diferentes para a análise dos sistemas em estudo.*

As principais abordagens utilizadas em sistemas são as seguintes (CURY, 2001):

- Redes de Petri com e sem temporização;
- Redes de Petri Controladas com e sem temporização;
- Álgebra de Dióides;
- Cadeias de Markov;
- Teoria de Filas;
- Processos Semi-Markovianos Generalizados (GSMP) e Simulação;
- Álgebra de Processos;
- Lógica Temporal e Lógica Temporal de Tempo Real;
- Teoria de Controle Supervisório (Ramadge–Wonham).

Os SEDs formam uma área de pesquisa de intensa atividade e desafios, aparecendo nos domínios da manufatura, robótica, tráfego de veículos, logística (transporte e armazenagem de bens, organização e entrega de serviços) e redes de computadores e de comunicação. Estas aplicações requerem controle e coordenação para assegurar o fluxo regular dos eventos (RAMADGE & WONHAM, 1989).

### 2.1.2 Sistemas a Eventos Discretos Controlados

Segundo Tsitsiklis (1987), *um sistema a eventos discretos é um sistema dinâmico a tempo discreto tal que, para cada estado, existe um número de diferentes transições que podem ocorrer. Assume-se, ainda, que é possível uma ação de controle feita através de um supervisor que possa proibir certas transições de ocorrerem, em um dado ponto qualquer no tempo. Este supervisor deve satisfazer certas especificações. Na literatura, as especificações que devem ser consideradas correspondem a requisições para que o supervisor proíba a ocorrência de certas seqüências de eventos, ao mesmo tempo que permite que outras seqüências de eventos ocorram.*

Seja $\Sigma$ o conjunto de todos os eventos de um SED. Este conjunto pode ser particionado em 2 subconjuntos disjuntos, tal que $\Sigma = \Sigma_c \cup \Sigma_{nc}$, onde $\Sigma_c$ é o conjunto de todos os eventos controláveis do SED e $\Sigma_{nc}$ o conjunto de todos os eventos não controláveis do SED. Os eventos controláveis são os eventos que podem ser desabilitados a qualquer momento por um agente externo, enquanto os eventos não controláveis são aqueles em que o agente de controle não tem influência, tais como quebras de máquinas em um sistema de manufatura, perturbação externa, perda de informação em canais de comunicação, etc. Desta forma define-se:

**Definição 2.1 Eventos Controláveis** – *um evento em um SED é dito ser controlável se a sua ocorrência pode ser inibida por uma ação externa.*

**Definição 2.2 Eventos Não Controláveis** – *um evento em um SED é dito ser não controlável se a sua ocorrência não pode ser inibida por uma ação externa.*

As entradas de controle são formadas pelo conjunto de eventos que se quer habilitar em um determinado estado do sistema. Note que as entradas de controle não devem tentar inibir eventos não controláveis. Por este motivo, uma entrada de controle é considerada válida somente se contiver o conjunto de eventos não controláveis, isto é, os eventos não controláveis estão necessariamente habilitados sempre.

Pode-se definir SEDs controlados da seguinte forma:

**Definição 2.3 Sistemas a Eventos Discretos Controlados** – *um sistema a eventos discreto controlado (SEDC) é um SED tal que o conjunto de todos os seus eventos $\Sigma$ pode ser particionado em dois subconjuntos disjuntos $\Sigma_c$ e $\Sigma_{nc}$, com $\Sigma_c$ sendo o conjunto dos eventos controláveis e $\Sigma_{nc}$ o conjunto dos eventos não controláveis, tal que $\Sigma_c \cup \Sigma_{nc} = \Sigma$ e $\Sigma_c \cap \Sigma_{nc} = \oslash$.*

Por uma questão de completude serão definidos a seguir os sistemas a eventos discretos com eventos observáveis.

Seja $\Sigma$ o conjunto de todos os eventos de um SED. Este conjunto também pode ser particionado em 2 subconjuntos disjuntos $\Sigma = \Sigma_o \cup \Sigma_{no}$ (da mesma forma que os conjuntos $\Sigma_c$ e $\Sigma_{nc}$), onde $\Sigma_o$ é o conjunto de todos os eventos observáveis do SED e $\Sigma_{no}$ o conjunto de todos os eventos não observáveis do SED. Os eventos observáveis são os eventos que podem ser avaliados por um agente externo, enquanto os eventos não observáveis são aqueles que não podem ser observados. As principais causas para não observabilidade dos eventos são as limitações dos sensores anexados ao sistema e a distribuição natural de alguns sistemas onde eventos de um local não podem ser vistos de outro local (CASSANDRAS & LAFORTUNE, 1999). Desta forma define-se:

**Definição 2.4 Eventos Observáveis** – *um evento em um SED é dito ser observável se as suas ocorrências puderem ser diretamente observadas ou avaliadas.*

**Definição 2.5 Eventos Não Observáveis** – *um evento em um SED é dito ser não observável se as suas ocorrências não puderem ser diretamente observadas ou avaliadas.*

É interessante notar que não existe uma relação direta entre eventos controláveis e eventos observáveis, isto é, um evento pode ser não observável, mas controlável, assim como pode ser observável e não controlável (CASSANDRAS & LAFORTUNE, 1999).

Em um sistema a eventos discretos, um supervisor ou controlador é um dispositivo que deve, em uma estrutura hierárquica, ser capaz de processar comandos enviados de níveis mais altos,

para sincronizar as atividades dos equipamentos abaixo deles, e informar seu estado. O supervisor deve ainda satisfazer um conjunto de restrições. Estas restrições podem estar relacionadas à capacidade de manufatura finita, taxas de produção, taxas de falhas e de reparo, disponibilidade de matéria-prima, precedência de tarefas, ordens de serviços, assim como outros parâmetros do sistema (DESROCHERS & AL-JAAR, 1995).

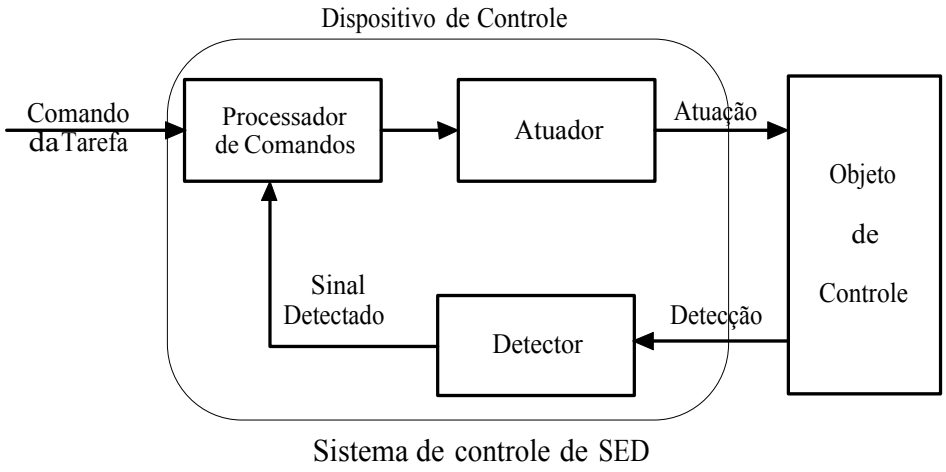



**Figura 2.2**: Diagrama Conceitual Básico de Controle de SED

O contexto de controle que se considera neste trabalho, pode ser resumido da seguinte forma. Supõe-se inicialmente, que um procedimento pré-estabelecido é conhecido, correspondendo aos inícios periódicos das tarefas do sistema, isto é, são conhecidos os instantes de início de cada tarefa e o período com que este inícios devem se repetir. Este procedimento pré-estabelecido é visto como um "sinal de referência"ou "procedimento de referência"ou ainda "ciclo de referência"para o sistema. Na ausência de perturbações, a saída do sistema deve corresponder ao procedimento de referência.

É comum entretanto que perturbações ocorram durante a operação do sistema, levando-o a um estado que não corresponde ao procedimento de referência. Isso pode acontecer, por exemplo, quando uma tarefa exige um tempo de execução além do previsto, tendo como consequências a inviabilização de partes do procedimento de referência devido ao aparecimento de conflitos e o aumento do período de operação do sistema.

Neste caso, um controlador reativo, isto é, um controlador capaz de responder a perturbações, teria como função determinar os intantes de início de cada tarefa, visando a recuperação do ciclo de referência. A partir do conhecimento destes instantes de início, o controlador determinaria a habilitação ou inibição dos eventos controláveis. Este controlador agiria portanto em malha fechada, isto é, utilizando as informações de saída do sistema constituídas pelos instantes em que as tarefas realmente ocorreram para a determinação dos instantes de início das tarefas nos ciclos seguintes. Além disso, operaria num intervalo transitório entre a ocorrência da perturbação e a retomada do procedimento em regime permanente, determinado pelo ciclo de referência.

Este contexto de controle, ilustrado na figura 2.2, é explorado nos trabalhos de Noronha &

Santos-Mendes (2000), Noronha & Santos-Mendes (2002), e Leandro *et. al.* (2002) e define um quadro geral no qual o presente trabalho deve ser considerado. Entretanto deve ficar claro que o objetivo aqui é unicamente determinar o procedimento de referência para um sistema a eventos discretos com comportamento periódico. Este procedimento será obtido a partir do escalonamento cíclico das tarefas do sistema.

### 2.1.3 Sistemas de Manufatura

É usual no tratamento dos SEDs a utilização de uma terminologia vinda da Teoria de Filas e também, utilizada em Simulação de Sistemas. Esta terminologia será adotada neste trabalho e é mostrada a seguir:

**Sistemas de Filas -** Os três elementos básicos de um sistema de filas são:

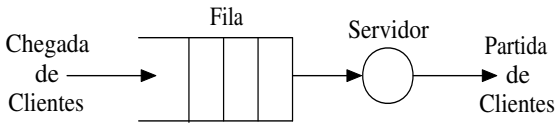



**Figura 2.3**: Sistema de Fila Simples

1. As entidades que devem esperar pelo uso dos recursos. Geralmente, estas entidades são denominadas *clientes*. Ex.: pessoas (esperando em um banco); mensagens transmitidas através de um canal de comunicação; tarefas ou processos executados num sistema de manufatura ou de computação.
2. Os recursos pelos quais se espera. Como em geral estes recursos fornecem alguma forma de serviço aos cliente, são denominados *servidores*. Ex.: pessoas (caixas de banco ou supermercado); canais de comunicação responsáveis pela transmissão de mensagens; processadores ou dispositivos periféricos em sistemas de computação; máquinas usadas na manufatura.
3. O espaço onde a espera se faz, denomina-se *fila*, ou em alguns casos, *buffers*. Ex.: filas de banco, supermercados; *buffers* de chamadas telefônicas ativas, ou processos executáveis.

Outro exemplo típico de Sistemas a Eventos Discretos são os sistemas de manufatura nos quais os clientes são as peças ou ítens, *pallets*; os servidores são máquinas, dispositivos de manuseio e de transporte (robôs, esteiras, . . .); e as filas são os armazéns ou *buffers*.

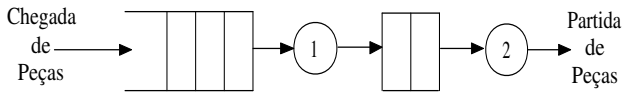



**Figura 2.4**: Linha de Transferência

A figura 2.4 ilustra o exemplo de uma Linha de Transferência com duas máquinas e um Armazém Intermediário de capacidade 2. Os eventos são: a chegada de peça, o término de serviço pela máquina 1 e a partida de peça da máquina 2.

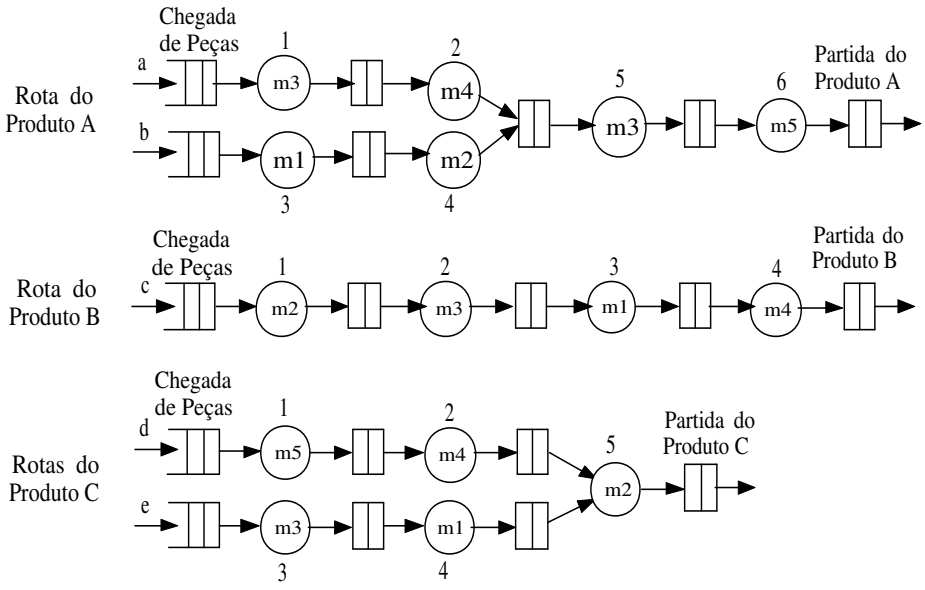



**Figura 2.5**: Rotas para a Manufatura de 3 Produtos na mesma Planta

A figura 2.5 mostra um exemplo de um sistema com 3 processos de confecção de produtos:

- Produto A - composto por dois tipos diferentes de subprodutos, a e b, cada um com sua própria rota até a máquina 3, onde é exigido que estes produtos sejam mesclados, formando o produto A. É necessário que a e b estejam liberados para que a máquina m3 possa processá-los, isto é, mesmo que a esteja liberado antes de b, a máquina m3 deverá esperar b estar disponível para começar o processamento deles. Logo, existe a necessidade de sincronização entre eles.

- Produto B - composto por um único produto que deve ser processado pelas máquina m2, m3, m1 em4, conforme estabelecido por sua rota.

- Produto C - composto por dois tipos diferentes de subprodutos, d e e, cada um com sua própria rota até a máquina m2. Depois de serem processados na máquina m2 tornam-se o produto C. Diferente do produto A, aqui a máquina m2 deve processar os produtos d e e em separado, havendo, portanto, uma concorrência entre eles para a ocupação da máquina m2.

Os sistemas de manufatura formam um caso particular de sistemas a eventos discretos e podem ser descritos como sendo sistemas formados por conjuntos de tarefas que interagem com conjuntos de recursos resultando em produtos (ZHOU & DICESARE, 1993). Estas tarefas são os processamentos de manufatura que incluem atividades nas máquinas, transporte de materiais e processamento de informação, que devem ocorrer para se produzir algo. Os recursos são os trabalhadores, máquinas, matérias-primas, por exemplo, que são requeridos para realizar estas tarefas. A programação do processamento do produto especifica as tarefas e os recursos em detalhes, incluindo as relações de precedência entre tarefas.

A seguir serão definidos alguns termos de Sistemas de Manufatura que serão muito utilizados no decorrer deste trabalho.

**Definição 2.6 Processo** *é um conjunto de eventos relacionados entre si por alguma condição de dependência imediata.*

Em geral a idéia de processo é usada para identificar um subsistema, em um SED, p.ex. um sistema de fila.

**Definição 2.7 Atividade ou Tempo Especificado** *é o intervalo de tempo entre dois eventos de um processo cuja duração não depende do estado do sistema, podendo ser determinística ou aleatória.*

Uma atividade pode ser, por exemplo, o tempo de duração de uma tarefa em uma máquina.

Serão consideradas neste trabalho, apenas as atividades com tempos determinísticos.

É usual que o intervalo de atividade seja também denominado de tempo de duração de uma tarefa ou tempo de duração de uma operação, ou simplesmente tarefa ou operação.

**Definição 2.8 Espera ou Atraso** *é o intervalo de tempo entre dois eventos de um processo cuja duração depende do estado do sistema.*

Uma espera pode ser, por exemplo, o tempo gasto na fila em um sistema de filas.

É usual no contexto de sistemas de manufatura utilizar o conceito de *job*, definido como a seguir:

**Definição 2.9 Job** *é uma seqüência finita de operações ordenadas linearmente segundo a dependência, onde cada operação tem um único antecessor (exceto a operação inicial) e um único sucessor (exceto a operação final).*

Cada operação de um *job* depende da disponibilidade do recurso associado a ela.

Em relação às definições anteriores um *job* é portanto uma classe particular de processos.

Em muitos casos em sistemas de manufatura o problema é encontrar a melhor ordem para a execução das tarefas alocadas nos recursos, isto é, o momento em que as atividades devem ocorrer. Em muitos sistemas de manufatura, esta ordem só é conseguida através de estudos específicos para o problema, e neste caso pode-se usar a Teoria de Escalonamento. Baker (1974) define o **Escalonamento** como sendo uma função de tomada de decisão, sendo tanto um processo através do qual determina-se um planejamento horário, quanto um corpo teórico, contendo uma coleção de princípios, de modelos, de técnicas e de conclusões lógicas que geram idéias para a função de Escalonamento.

Um dos problemas de Escalonamento mais conhecidos é o problema de job shop que pode ser descrito como a seguir: $n$ jobs *devem ser processados em* $m$ *diferentes máquinas, cada operação dos* jobs *deve ser executada em uma única máquina previamente estipulada e com uma duração fixa.*

No contexto deste trabalho o conceito de *job* será generalizado de modo a permitir a modelagem de sistemas mais complexas.

**Definição 2.10 Job Generalizado** *é uma seqüência de operações linearmente ordenadas, com cada operação dependendo de seu antecessor e dos recursos a ela associados.*

Note-se que nesta definição admite-se a existência de mais de um antecessor e de mais de um sucessor para a mesma operação.

Por exemplo, a rota de produção do produto A da figura 2.5 define um único processo, composto por dois *jobs* generalizados, sendo um deles formado pelas operações $\{1, 2, 5, 6\}$ e o outro pelas operações $\{3, 4, 5, 6\}$.

Um problema de escalonamento que engloba os *jobs* generalizados é o Problema de Escalonamento Cíclico (PEC), que é um problema de escalonamento em um sistema com produção repetitiva e que pode ser definido da seguinte forma (DAUSCHA *et al.*, 1985):

**Definição 2.11 Problema de Escalonamento Cíclico –** *Um dado conjunto finito de operações deve ser escalonado em um fluxo contínuo e periódico, isto é, todas as operações devem ser repetidamente executadas de forma que o intervalo de tempo entre dois pontos de ativação sucessiva de uma mesma operação é uma constante Z para todas as operações. Esta constante Z recebe o nome de comprimento do período (ou do ciclo) e uma atribuição dos tempos de ativação das operações por escalonamento cíclico.*

Como o PEC trata de sistemas cíclicos que são o objeto de estudos deste trabalho, este problema será estudado no capítulo 4.

No que se segue a palavra *job* será usada livremente para designar *job generalizado*.

## 2.2 Programação Linear Inteira

Segundo Hillier & Lieberman (1988), *São muitos os problemas reais que só têm sentido se as variáveis de decisão forem inteiras, por exemplo, frequentemente é necessário alocar pessoas, máquinas ou outros recursos a atividades, em quantidades inteiras. Esta restrição é difícil de ser manipulada matematicamente. Entretanto, tem-se conseguido algum progresso no desenvolvimento de procedimentos de solução para o caso de problemas de programação linear sujeitos a esta restrição adicional de que as variáveis de decisão tenham que ter valores inteiros.*

O grupo de problemas em programação matemática que envolvem variáveis inteiras recebem o nome de problemas de programação inteira. A programação inteira pode-se dividir em inteira pura (IP), caso estejam envolvidas apenas variáveis inteiras no problema, em inteira mista (MIP), caso existam variáveis inteiras e variáveis contínuas e em programação linear inteira mista (*mixed integer linear programming* – MILP),quando as restrições e a função–objetivo dos problemas são lineares.

A abordagem mais simples para problemas MILP é usar o método *simplex* (ignorando a restrição das variáveis serem inteiras) e, em seguida, arredondando os valores não inteiros para inteiros, na solução resultante. Embora em algumas vezes isso seja adequado, existem falhas nesta abordagem. Uma das falhas é que a solução ótima de programação linear *não é necessariamente ótima* depois do arredondamento, sendo frequentemente difícil saber de que maneira o arredondamento deve ser feito para que a factibilidade seja mantida (HILLIER & LIEBERMAN, 1988).

Mesmo que a solução ótima de programação linear seja arrendondada com sucesso, restará outra falha – não existe garantia de que esta solução arredondada seja a solução ótima inteira.

Segundo Hillier & Lieberman(1988), *por estas razões, a existência de um procedimento eficaz para a obtenção da solução ótima para problemas de programação linear inteira mista seria muito conveniente. Um considerável número de algoritmos foi desenvolvido para este propósito. Infelizmente, nenhum possui uma eficiência computacional que seja mesmo remotamente comparável à do método simplex (exceto em tipos especiais de problemas). Por isto esta continua sendo uma área ativa de pesquisa, e continuam a ser feitos progressos no desenvolvimento de algoritmos mais eficientes*.

Atualmente esta busca continua, por esta razão (e por não ser um dos objetivos da tese), esta tese não irá utilizar nenhum procedimento específico para a resolução da modelagem em MILP, usando um programa comercial que contém os algoritmos mais eficientes para a resolução de problemas inteiros.

## 2.3 Redes de Petri

Segundo Reisig (1992), *as redes de Petri são ferramentas com grande poder de modelagem gráfica, com fundamentos matemáticos firmemente incorporados que representam um sistema como um conjunto de entidades ativas e passivas interagindo. As entidades ativas são chamadas transições, dado que as transições são os entes que "disparam"e as passivas são chamadas de lugares, pois elas simplesmente definem as condições de disparo*.

Uma rede de Petri pode ser usada como ferramenta gráfica, para auxiliar na visualização de um problema. Fichas (ou marcações) são usadas nestas redes para simular a dinâmica e as atividades concorrentes do sistema, tornando mais fácil para os projetistas a compreensão a respeito de seus sistemas, facilitando a utilização das técnicas de análise. Uma rede de Petri pode ser usada também como um modelo matemático, configurando-se como equações de estado, equações algébricas e outros modelos matemáticos para controlar o comportamento dos sistemas.

O conceito de Rede de Petri teve sua origem na tese de doutorado de Carl Adam Petri, intitulada *Kommunikation mit Automaten (Comunicação com Autômatas)* (PETRI, 1962), criando-se assim uma abordagem teórica capaz de modelar e analisar sistemas de comunicação.

As redes de Petri incorporam as características tanto das máquinas de estado finito quanto de grafos direcionados bipartidos. Elas podem expressar transições de estado causadas por eventos, como também, atividades processadas em paralelo. É um modelo classificado como *estado-evento*, com cada evento possuindo pré-condições que vão permitir sua ocorrência e pós-condições decorrentes desta, as quais podem também ser pré-condições de outros eventos.

Redes de Petri são modelos formais de especificação e controle do fluxo de informações de sistemas discretos, por isso, suas áreas típicas de aplicação são os sistemas de tempo real, sistemas dinâmicos a eventos discretos, controle de manufatura flexível, robótica, sistemas automotivos, ambiente de multiprocessadores, protocolos de comunicação e de sincronização, sistemas distribuídos, sistemas interativos, programação lógica, redes neurais e controladores nebulosos (fuzzy) (GEROGIANNIS it et al., 1998).

### 2.3.1 Representação de Redes de Petri

Uma rede de Petri é representada por um grafo direcionado bipartido que permite modelar as propriedades estáticas de um sistema a eventos discretos, constituído de dois tipos de nós (PETERSON, 1981):

- as transições, que correspondem aos eventos que caracterizam as mudanças de estado do sistema e são simbolizadas por retângulos ou barras;
- os lugares, que correspondem às condições que devem ser certificadas para que os eventos ocorram e são simbolizados por círculos.

Arcos direcionados ponderados ligam os nós de tipos diferentes, isto é, transições a lugares e lugares a transições. Um arco direcionado não representa um componente do sistema, mas simboliza o relacionamento entre os componentes.

Os itens usados para representar a dinâmica do sistema, como por exemplo, um pedaço de informação ou de controle que fluem entre os lugares e através das transições são chamados de fichas. As fichas são representadas por pontos dentro dos lugares. A presença ou ausência de uma ficha em um lugar pode indicar, por exemplo, se uma condição associada a este lugar é verdadeira ou falsa (WANG, 1998).

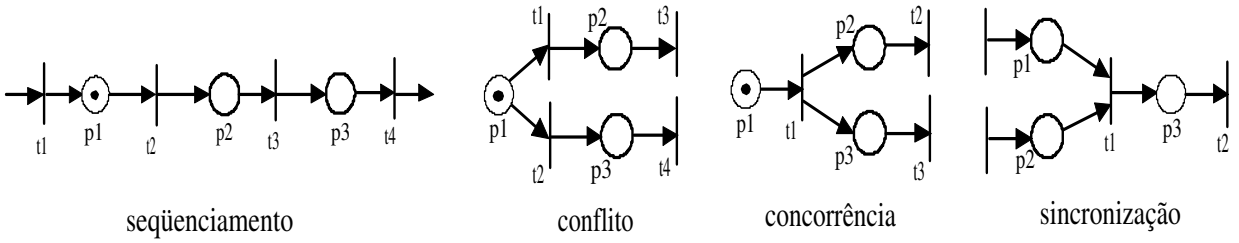



**Figura 2.6**: Relações de causalidade modeladas por Redes de Petri

A rede de Petri é um formalismo com grande poder de expressividade, que permite a modelagem de uma grande classe de sistemas dinâmicos a eventos discretos, representando com facilidade todas as relações de causalidade entre processos em situações de sequencialidade, conflito, concorrência e a sincronização como mostrado pela figura 2.6.

**Definição 2.12** *(PETERSON,1981) Uma rede de Petri com* $n$ *lugares e* $m$ *transições pode ser representada por quádrupla* $N =< P, T, I, O >$*, onde:*

- $P = \{p_1, p_2, \ldots, p_n\}$ *é um conjunto de lugares;*
- $T = \{t_1, t_2, \ldots, t_m\}$ *é um conjunto de transições, com* $P \cup T \neq \oslash$ *e* $P \cap T = \oslash$*;*

- $I : P \times T \rightarrow \mathbb{N}$, *é a função de entrada ou pré-incidência que especifica os arcos direcionados dos lugares para as transições;*
- $O : T \times P \rightarrow \mathbb{N}$ *é a função de saída ou pós-incidência que especifica os arcos direcionados das transições para os lugares.*

*onde* $\mathbb{N}$ *representa o conjunto dos números naturais.*

A maioria dos trabalhos em redes de Petri é baseada na definição formal das estruturas da rede de Petri mostrada pela definição 2.12. Entretanto, a representação gráfica de uma rede de Petri é muito mais útil para ilustrar os conceitos da teoria de rede de Petri.

Como dito anteriormente, um grafo de rede de Petri é um grafo direcionado bipartido contendo dois tipos de nós: os lugares e as transições. Arcos direcionados conectam lugares e transições, com alguns arcos direcionados dos lugares para as transições e outros das transições para os lugares. Um arco direcionado de um lugar $p_j$ para uma transição $t_i$ define $p_j$ como sendo um lugar de entrada de $t_i$, denotado por $I(p_j, t_i) = 1$. Um arco direcionado da transição $t_i$ para o lugar $p_j$ define $p_j$ como sendo um lugar de saída de $t_i$, denotado por $O(p_j, t_i) = 1$. Se $I(p_j, t_i) = k$ (ou $O(p_j, t_i) = k$), existem $k$ arcos direcionados paralelos conectando o lugar $p_j$ à transição $t_i$ (ou conectando a transição $t_i$ ao lugar $p_j$). Normalmente, em uma representação gráfica, arcos paralelos conectando lugares (transições) às transições (lugares) são representados por um único arco direcionado etiquetado com sua multiplicidade, ou peso $k$. Um círculo que contenha um ponto representa um lugar que contêm uma ficha. A figura 2.7 mostra um exemplo de rede de Petri.

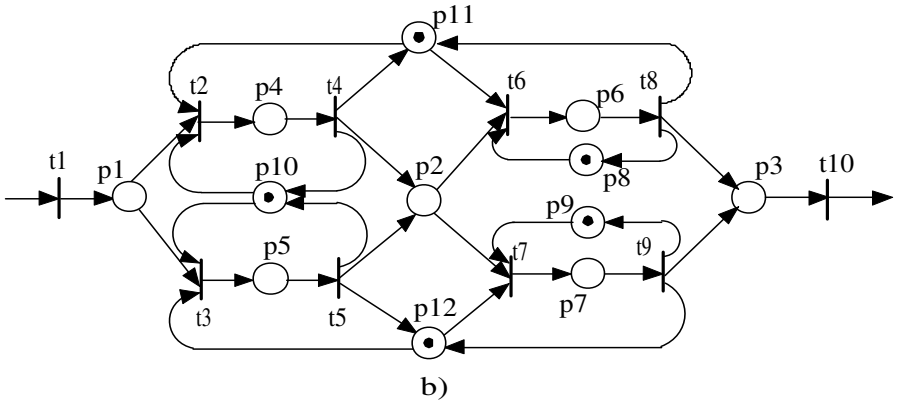



**Figura 2.7**: Rede de Petri

Uma marcação é uma atribuição de fichas aos lugares de uma rede de Petri. Uma ficha é um conceito primitivo em redes de Petri (assim como lugares e transições). Essas fichas residem nos lugares. O número e a posição das fichas mudam durante a execução de uma rede de Petri (PETERSON, 1981).

**Definição 2.13** *Uma marcação* **m** *de uma rede de Petri* $N = \langle P, T, I, O \rangle$ *é uma função do conjunto de lugares ao conjunto de números inteiros não negativos* $\mathbb{N}$*, isto é,* $m : P \rightarrow \mathbb{N}$.

**Definição 2.14** *Uma* **rede de Petri marcada** *é uma dupla* $\langle N, m_0 \rangle$*, onde* $N$ *é uma rede de Petri e* $m_0$ *é uma marcação inicial.*

Pode-se estudar o comportamento dinâmico de um sistema modelado através da mudança na distribuição de fichas nos lugares. O movimento de fichas entre lugares é controlado pelas transições da RdP. A posição das fichas definem o estado do sistema, definindo situações tais como condições satisfeitas, itens em um armazenador (*buffer*), o número de servidores livres, recursos disponíveis e entidades em filas. A distribuição de fichas sobre a RdP é definida pela marcação da RdP. É a marcação da RdP que define o estado corrente do sistema modelado. As regras a seguir são usadas para comandar o fluxo das fichas (ZURAWSKI & ZHOU, 1994):

- *Regra de Habilitação*: Uma transição $t \in T$ é dita estar habilitada se cada lugar de entrada $p$ de $t$ contêm pelo menos um número de fichas igual ao peso do arco direcionado conectando $p$ a $t$, isto é, $m(p) \geqslant I(p,t)\ \forall p \in P$.

- *Regra de Disparo*:

  - Um disparo de uma transição habilitada $t$ remove de cada lugar de entrada $p$ o número de fichas igual ao peso do arco direcionado conectando $p$ a $t$. Ela deposita em cada lugar de saída $p$ o número de fichas igual ao peso do arco direcionado conectando $t$ a $p$.
  - A habilitação da transição $t$ gera para esta transição a expectativa de disparo, cabendo ao controlador da rede o direito de disparar as transições habilitadas na ordem em que achar conveniente.

O disparo de $t$ em $m$ gera uma nova marcação, isto é:

$$m(p)' = m(p) + O(p,t) - I(p,t),\ \forall p \in P. \tag{2.1}$$

A marcação $m'$ é dita ser alcançável de $m$. O conjunto de alcançabilidade da rede de Petri $N$ é o conjunto de todas as marcações alcançáveis por $N$ a partir de $m_0$ e é denotado por $R(N, m_0)$.

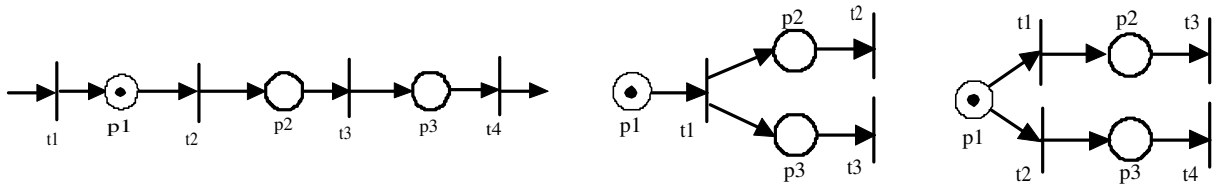
Somente transições habilitadas podem disparar, por esta razão, o número de fichas em cada lugar permanece sempre não negativo quando uma transição é disparada. Uma transição disparada nunca poderá remover uma ficha que não está lá.

Uma transição sem nenhum lugar de entrada é chamada de **transição de origem ou de entrada** e uma transição sem nenhum lugar de saída é denominada de **transição de destino ou saída**. Note que uma transição de origem está incondicionalmente habilitada, e que uma transição de destino consome fichas, mas não as produz.

Um par composto por um lugar $p$ e uma transição $t$ é chamado de ciclo próprio (*self loop*), se $p$ é tanto o lugar de entrada e o lugar de saída de $t$. Uma Rede de Petri é dita ser *pura* se ela não contêm ciclos próprios. As redes de Petri que serão estudadas neste trabalho são as redes de Petri puras.

### 2.3.2 Topologias características em redes de Petri

As redes de Petri modelam com eficiência as características mais comuns mostradas pelas atividades dos Sistemas Dinâmicos a Eventos Discretos (SEDS), tais como concorrência, sincronização, sequenciamento de eventos e conflito (ver seção 2.1). O conflito simboliza uma tomada de decisão, como por exemplo no caso de uma exclusão mútua. A seguir são introduzidas algumas topologias de redes de Petri que representam características usuais de sistemas dinâmicos a eventos discretos. Estas topologias serão usadas na seção 2.4 para redefinir o conceito de processo. Por sua vez este novo conceito de processo constituírá o elemento fundamental para a modelagem dos sistemas a eventos discretos periódicos (i.e., sistemas cíclicos) proposta no capítulo 4.




a) Execução Sequencial b) Concorrência c) Conflito

**Figura 2.8**: Algumas topologias em redes de Petri

Seja $\pi(j)$ o conjunto de todos os elementos predecessores do elemento $j$ e $\lambda(j)$ o conjunto dos elementos sucessores(ver seção A) do elemento $j$, define-se:

**Definição 2.15** *Execução Sequencial – A execução sequencial se caracteriza por um conjunto de transições* $(t_1, t_2, \ldots, t_p)$, $p > 1$, *que devem disparar em seqüência, começando por* $t_1$ *(transição inicial) e terminando por* $t_p$ *(transição final).*

A fig. 2.8a ilustra um exemplo. Neste caso, a transição $t_4$ é a transição final e só poderá disparar após o disparo de $t_3$, que por sua vez só disparará depois de $t_2$.

**Definição 2.16** *Concorrência – A concorrência se caracteriza por um conjunto de transições* $(t_1, t_2, \ldots, t_p)$, $p > 1$, *que podem disparar independentemente umas da outras.*

Na fig. 2.8b as transições $t_2$ e $t_3$ são concorrentes. A concorrência é um importante atributo de interações de Sistemas Dinâmicos a Eventos Discretos.

**Definição 2.17** *Conflito – Um conflito surge em um conjunto de transições* $(t_1, t_2, \ldots, t_p)$, $p > 1$, *quando ao existir duas ou mais dessas transições habilitadas a ocorrer, a ocorrência de qualquer uma delas desabilita todas as outras.*

A fig. 2.8c exemplifica este caso, com o disparo da transição $t_1$ desabilitando a transição $t_2$, ou vice-versa.

Neste trabalho, apenas dois tipos de sincronização serão considerados.

**Definição 2.18** *Sincronização – A sincronização pode ocorrer de duas formas, join e fork, que são descritas a seguir:*

1. *join – ocorre em um conjunto de transições* $(t_1, t_2, \ldots, t_p)$, $p > 1$, *que podem disparar concorrentemente umas das outras, mas estão associadas à mesma transição sucessora (através de seus lugares de saída) que não pertence a este conjunto. Neste caso, apenas o disparo de todas as transições deste conjunto poderá habilitar a transição sucessora.*

2. *fork – ocorre em um conjunto de transições* $(t_1, t_2, \ldots, t_p)$, $p > 1$, *que podem disparar concorrentemente umas das outras, mas estão associadas à mesma transição predecessora (através de seus lugares de entrada) que não pertence a este conjunto. Neste caso, apenas o disparo da transição predecessora poderá habilitar todas as transições deste conjunto.*

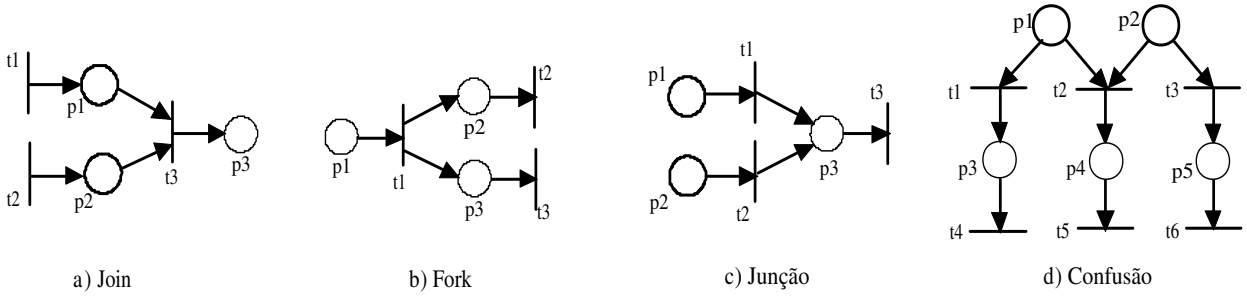



**Figura 2.9**: Algumas topologias de redes de Petri

A fig. 2.9a mostra um exemplo de *join* e a fig. 2.9b mostra um exemplo de *fork*.

**Definição 2.19** *Junção – A junção se caracteriza por um conjunto de transições* $(t_1, t_2, \ldots, t_p)$, $p > 1$, *que são concorrentes entre si, tal que o disparo de qualquer uma das transições deste conjunto pode habilitar uma outra transição que não pertence a este conjunto, mas que é dependente de todas elas.*

A fig. 2.9b mostra um exemplo, onde a transição $t_3$ poderá ocorrer após o disparo tanto de $t_1$ quanto de $t_2$.

**Definição 2.20** *Confusão – A confusão é a existência de concorrência e de conflito em uma mesma transição.*

Na fig. 2.9c, a transição $t_1$ é concorrente a $t_3$, mas ambas estão em conflito com $t_2$, caracterizando a confusão.

**Definição 2.21** *Exclusão Mútua – A estrutura de exclusão mútua é dada por um conjunto de transições* $(t_1, t_2, \ldots, t_n)$, $n > 1$, *que são saídas de um mesmo lugar* $p$, *podem disparar concorrentemente, mas não podem disparar simultaneamente, sendo portanto mutuamente exclusivas. Cada uma destas transições é entrada de alguma sub-rede*[1]*, cujas transições de saída também são transições de entrada de* $p$.

É usual, ainda, na definição de exclusão mútua, determinar que o lugar que represente a exclusão mútua possua exatamente uma ficha em sua marcação inicial, de forma que realmente seja impossível o disparo simultâneo de suas transições de saída (ZHOU & DICESARE, 1993). Porém neste trabalho, os lugares que satisfizerem a definição 2.21 e contiverem uma ou mais fichas em suas marcações iniciais serão chamados de lugar de exclusão mútua.

A fig. 2.10 ilustra dois exemplos. No exemplo da figura 2.10(a) as transições $t_3$ e $t_4$ são mutuamente exclusivas, isto é, os processos relacionados a estas transições partilham o uso do mesmo recurso (simbolizado pelo lugar $p_5$) mas seu uso não pode ser simultâneo, pois existe apenas uma ficha em $p_5$. No exemplo da figura 2.10(b) as transições $t_1$, $t_4$ e $t_5$ são mutuamente exclusivas, o lugar $p_2$ é o lugar compartilhado e as transições $t_2$, $t_3$ e $t_6$ são suas transições de entrada.

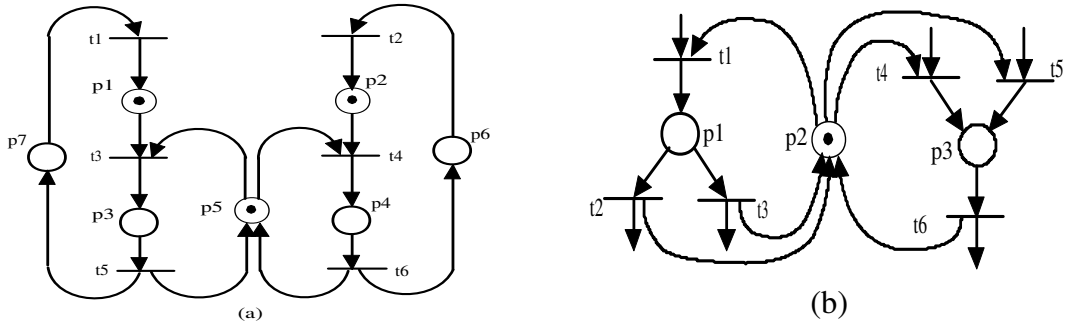



**Figura 2.10**: Exclusão Mútua

## 2.4 Processos

O conceito de processo é um dos conceitos mais importantes no estudo dos sistemas a eventos discretos. Dado que um processo é um conjunto de eventos relacionados entre si por alguma condição de dependência imediata (definição 2.6 da seção 2.1.2), ele pode ser descrito por uma rede de Petri.

Para facilitar o entendimento deste conceito, considere-se o sistema de manufatura multiproduto cujas rotas são mostradas pela figura 2.5, este sistema pode ser modelado pela rede de Petri mostrada pela figura 2.11. Cada rota em execução corresponde a uma seqüência de transições que devem ser disparadas de acordo com as disponibilidades de suas máquinas e as disponibilidades de seus lugares de entrada. Como o disparo de uma transição corresponde a ocorrência de um evento, cada rota representada por uma rede de Petri é um processo e o conjunto de rotas também é um processo.

[1] podendo ser formada de um único lugar

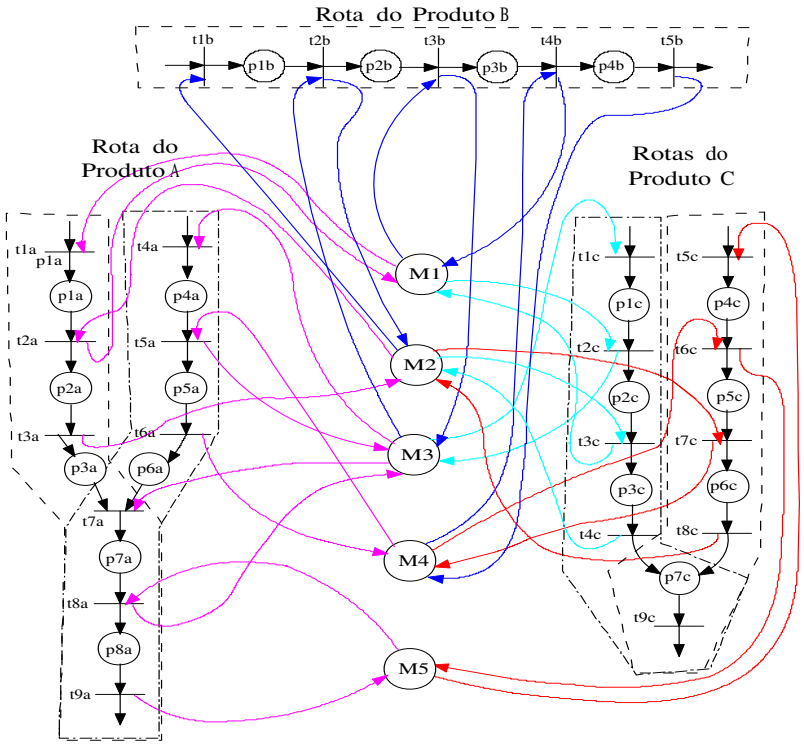



**Figura 2.11**: Rede de Petri das rotas mostradas na figura 2.5

As rotas modeladas pela rede de Petri e mostradas pela figura 2.11, possuem topologias diferentes, apesar disto todas podem ser denominadas de processos. Através deste exemplo é possível notar que enquanto a rota do produto A exige uma sincronização, a rota do produto C exibe uma disputa entre os seus subprodutos pela utilização da máquina 2. As diversas topologias de rede apresentadas anteriormente tornam possível discriminar estes tipos de processos, redefinindo-os em termos de Redes de Petri. A idéia central é que os processos assim definidos quando devidamente combinados permitem a descrição de uma classe ampla de sistemas a eventos discretos periódicos. Estas definições serão necessárias mais tarde na modelagem de Sistemas Cíclicos no cap. 4. Sendo $\pi(t)$ o conjunto de transições predecessoras da transição $t$ e $\lambda(t)$ o conjunto de transições sucessoras da transição $t$ pode-se definir:

**Definição 2.22 Processo Simples (PS) -***Um processo simples* J *é modelado por uma seqüência de transições* $(t_1,t_2,\ldots,t_p)$, $p > 1$, *que devem disparar em seqüência, com a transição* $t_1$ *sendo a transição inicial e* $t_p$ *a transição final desta seqüência .*

Note que esta é uma definição similar à da Execução Sequencial da seção 2.3.2. Os processos simples representam *jobs* generalizados (ver definição 2.10) em um sistema de manufatura.

No sistema de manufatura ilustrado pela figura 2.5, a rota do produto B é um *processo simples* e pode ser representada pela rede de Petri da figura 2.12.

**Definição 2.23 Processo com Sincronismo (PCS) -** *Um processo com sincronismo é um processo*

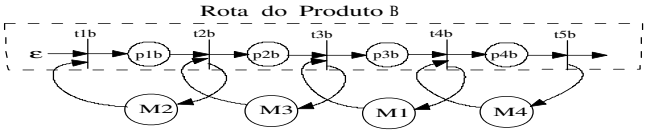



**Figura 2.12**: Rede de Petri da rota do produto B do exemplo 2.5

*descrito por uma rede de Petri que possua lugares com exatamente uma transição de entrada e uma de saída e de tal forma que todas as transições iniciais precedam todas as transições finais.*

A topologia típica deste processo é constituída pela combinação das estruturas de execução sequencial (definição 2.15), de concorrência (definição 2.16) e de sincronismo (definição 2.18). No sistema de manufatura ilustrado pela figura 2.5, a rota do produto A é um processo com sincronismo e pode ser representada pela rede de Petri da figura 2.13. A transição t7a representa o sincronismo entre os dois processos simples {t1a, t2a, t3a} e {t4a, t5a, t6a}.

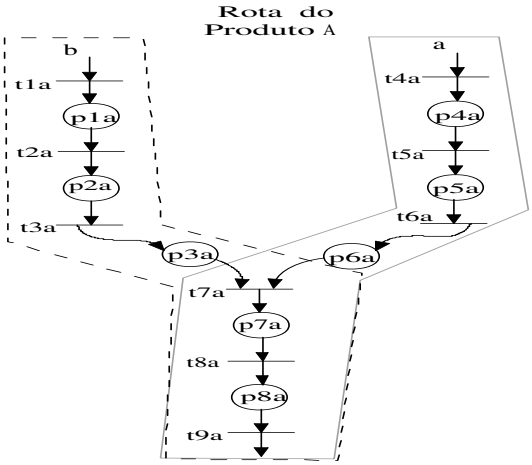



**Figura 2.13**: Rede de Petri da rota do produto A do exemplo 2.5

**Definição 2.24 Processos com Alternância (PCA) -** *Um processo com alternância é um processo descrito por uma rede de Petri que possua transições com exatamente um lugar de entrada e um de saída e de tal forma que todas as transições iniciais precedam todas as transições finais.*

A topologia típica deste processo é constituída pela combinação das estruturas de execução sequencial (definição 2.15), de conflito (definição 2.17), de junção (definição 2.19), e de confusão (definição 2.20). No sistema de manufatura ilustrado pela figura 2.5, a rota do produto C é um processo com alternância e pode ser representada pela rede de Petri da figura 2.14. O lugar p7c representa uma junção entre os dois processos simples, formados por {t1c, t2c, t3c, t4c} e {t5c, t6c, t7c, t8c}.

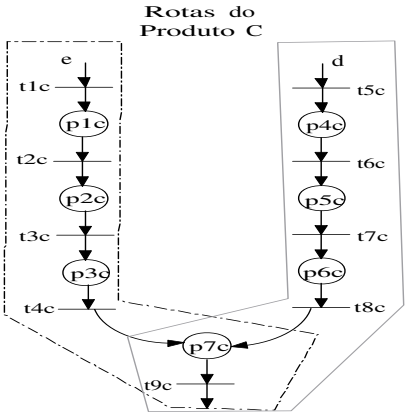



**Figura 2.14**: Rede de Petri da rota do produto C do exemplo 2.5

**Definição 2.25 Processos com Compartilhamento (PCC) -** *Um processo com compartilhamento é um processo descrito por uma rede de Petri com pelo menos um lugar representando a condição de exclusão mútua (definição 2.21).*

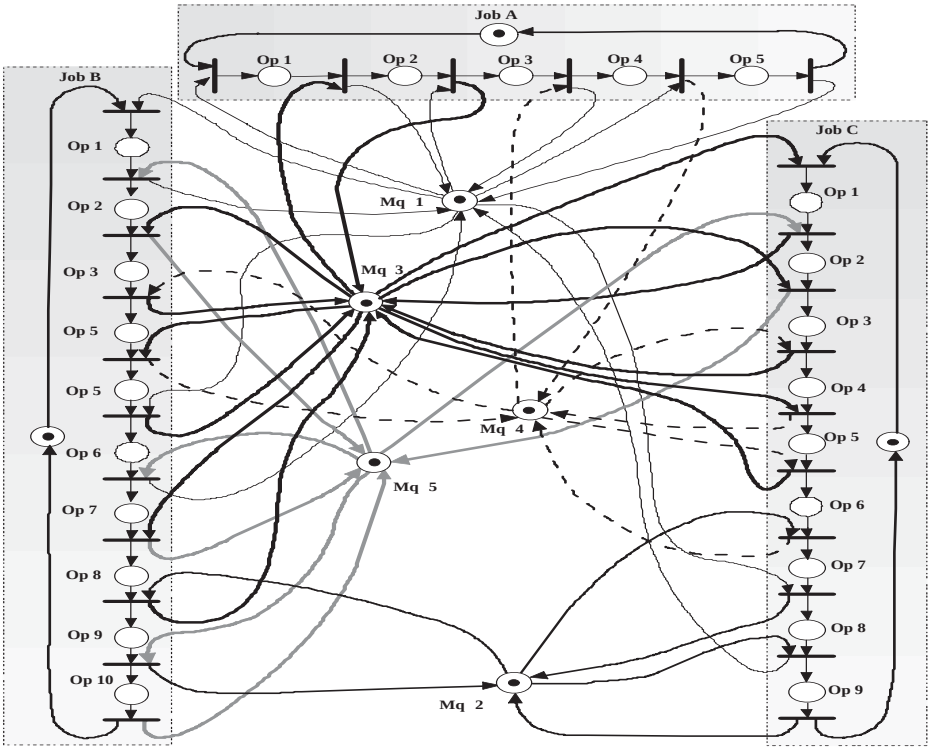



**Figura 2.15**: Rede de Petri de Processos com Compartilhamento

A figura 2.15 ilustra um exemplo de processos com compartilhamento em redes de Petri. Esta é a rede de Petri do exemplo B.1 que contêm três jobs que compartilham cinco máquinas mostrado

no apêndice B.

Note que os processos com compartilhamento podem conter processos com sincronismo, com alternância e/ou processos simples em sua topologia.

Note também que de acordo com a definição de processo simples, os processos com alternância, os com sincronismo e os com compartilhamento são combinações de processos simples através das estruturas mostradas na seção 2.3.2. Para distinguir-se os processos simples que compõe estes processos, daqueles que não pertencem a nenhum processo composto, defini-se:

**Definição 2.26** *Processo Ordinário (PO) – Um processo ordinário* $J$ *é modelado por uma seqüência de transições* $(t_1, t_2, \ldots, t_p)$, $p > 1$, *que devem disparar em seqüência, com a transição* $t_1$ *sendo a transição inicial e* $t_p$ *a transição final desta seqüência , e de forma que* $t_1$ *seja a única transição inicial e* $t_p$ *seja a única transição final da seqüência .*

Por esta definição, um processo ordinário é um processo simples que não faz parte de nenhum processo composto (processo com sincronismo, com alternância, com compartilhamento).

Levando em consideração a definição 2.9 para *job* e a definição 2.10 para *job generalizado*, *jobs* são modelados por processos ordinários e *jobs generalizados* são modelados por processos simples.

## 2.5 Objetivos da tese

O objeto de estudo deste trabalho são os SED Periódicos (SEDP) definidos a seguir:

**Definição 2.27 Comportamento Periódico de um Evento** – *Seja* $\alpha$ *um evento associado a um SED. Diz-se que* $\alpha$ *tem um comportamento* $Z$*–periódico se existir um instante* $t_0$ *e um intervalo* $Z$ *tais que* $\forall t \geqslant t_0$, *se* $\alpha$ *ocorrer em* $t$ *então* $\alpha$ *ocorrerá em* $t + Z$.

**Definição 2.28 Sistema a Eventos Discretos Periódicos (SEDP)** – *Um sistema de eventos discretos controlado é dito* $Z$*–periódico se existir uma lei de controle que leve todos os eventos do sistema a um comportamento periódico.*

Neste trabalho, considera-se que o objetivo de controle para um SEDP é fazê-lo funcionar de modo regular com comportamento $Z$–periódico, com $Z$ sendo estipulado através da otimização do escalonamento cíclico do SEDP, de modo que o valor de $Z$ seja o menor possível, sem levar em consideração possíveis perturbações que possam ocorrer no sistema.

Conforme discutido anteriormente, o estabelecimento de uma lei de controle para um SED supõe a determinação do conjunto de eventos habilitados para cada estado, ou seqüência de eventos observados.

A determinação de um comportamento $Z$–periódico será feita através de um escalonamento cíclico que leve em conta todas as restrições do sistema.

De modo geral, o escalonamento cíclico permite determinar as regras para operação do sistema apenas em *regime permanente*, não constituindo uma lei de controle propriamente dita. Entretanto,

é imediato observar que, se as restrições do sistema forem observadas na determinação do escalonamento cíclico, então os instantes de início de atividades calculados correspondem aos instantes em que os correspondentes eventos devem ser habilitados.

Desse modo, delimita-se o escopo deste trabalho como sendo a determinação de um escalonamento que estabeleça um regime periódico para as atividades do sistema. Os problemas de *start-up* e de *respostas a perturbações na planta* não serão abordados neste trabalho.

As redes de Petri serão os modelos adotados neste trabalho para a descrição dos sistemas a eventos discretos. Esta escolha se deve à facilidade que esta ferramenta possui para a descrição destes sistemas. Entretanto, deve-se ressaltar que o problema de síntese de redes de Petri não será tratado neste trabalho. Por esta razão, considera-se que os sistemas a serem tratados são previamente modelados.

Para a resolução do escalonamento cíclico será adotada uma modelagem em programação linear inteira mista (MILP). Embora a modelagem em MILP não seja dependente das redes de Petri no que concerne à resolução do escalonamento cíclico, a descrição do sistema feita pelas redes de Petri permite verificar se o sistema possui algumas propriedades necessárias a esta resolução.

## 2.6 Metodologia da Tese

A metodologia está fundamentada na identificação dos processos nos sistemas. A partir dos processos é possível formular um problema de escalonamento e resolvê-lo, conforme será detalhado no cap. 4.

As etapas da solução do problema são, a partir da rede de Petri:

1. verificação da consistência do sistema, que torna possível a modelagem da periodicidade do sistema e a decomposição da RdP em processos;
2. identificação dos processos, que é necessária à formulação em MILP;
3. modelagem em Programação Linear Inteira Mista (MILP):
   - formulação do problema de escalonamento cíclico,
   - resolução do problema de escalonamento cíclico e busca do ciclo ótimo.

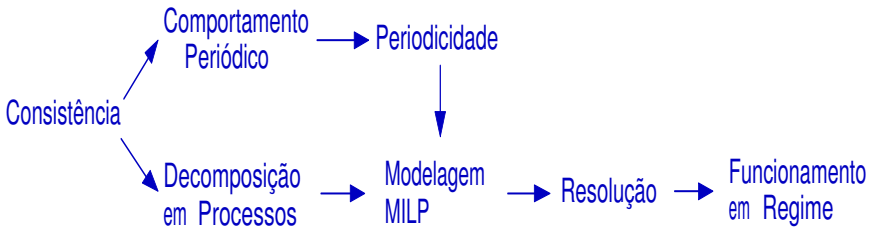



**Figura 2.16**: Metodologia da Tese

A figura 2.16 ilustra esta metodologia.

No capítulo 3 serão vistas as propriedades fundamentais das redes de Petri que permitem a solução do escalonamento cíclico.

Desse modo a própria metodologia delimita a classe de sistemas tratáveis neste trabalho, a saber:

1. estar contida na classe de sistemas a eventos discretos, modeláveis por redes de Petri, constituídas por combinações dos processos caracterizados na seção 2.4;
2. apresentar comportamento periódico verificado através das propriedades de conservação e de consistência, descritas na seção 3.2.2.
3. apresentar as propriedades de vivacidade, limitabilidade e reversibilidade a serem vistas na seção 3.2.

Caso a 1ª condição acima não seja satisfeita em muitos casos são possíveis transformações na rede de Petri de modo a permitir a identificação dos processos descritos na seção 2.4. O capítulo 5 trata de tais transformações.

É importante observar que os tipos de processos na seção 2.4 não são exaustivos em relação aos sistemas a eventos discretos. Entretanto estes tipos constituem um conjunto abrangente, descrevendo um leque amplo de situações práticas. A abrangência desta caracterização define a classe de sistemas tratáveis pelo método apresentado neste trabalho.

# 3

# Redes de Petri em Sistemas a Eventos Discretos

Este capítulo tem como objetivo mostrar as classes, subclasses, principais propriedades e características de redes de Petri que são usadas neste trabalho para modelar sistemas a eventos discretos com comportamento cíclico.

## 3.1 Classes e Principais Características

Na literatura existem muitas variantes do modelo original de Redes de Petri. Estas variantes nasceram da necessidade de adaptação das RdP à especificidade da aplicação para as quais a sua utilização era desejada. Por esta razão, considerou-se interessante efetuar um levantamento que apesar de necessariamente incompleto, permitisse uma sistematização dos tipos de Redes de Petri que mais frequentemente surgem na literatura. Com este levantamento, será possível localizar melhor as redes de Petri que poderão ser tratadas pela nossa modelagem.

As Redes de Petri podem ser divididas em três classes: Redes de Petri Ordinárias, Abreviações (ou Reduções) e Extensões.

Numa RdP Ordinária todos os arcos têm o mesmo peso (de valor 1), as fichas são de um mesmo tipo (isto é, não são diferenciadas), a capacidade dos lugares é infinita, nenhum tempo é envolvido e o disparo de uma transição só poderá ocorrer se cada lugar precedente tiver pelo menos uma ficha. No cap. 2 apenas redes de Petri ordinárias foram estudadas.

As abreviações correspondem a representações simplificadas que têm por finalidade facilitar a representação gráfica e possuem o mesmo poder de modelagem das RdP Ordinárias, de tal forma que sempre é possível encontrar uma rede de Petri Ordinária correspondente a uma rede de Petri abreviada. Dentro desta classe estão as RdP Generalizadas, as RdP com Capacidade Finita, e as RdP Coloridas. Neste caso todas as propriedades de uma RdP Ordinária são mantidas com umas poucas adaptações.

As extensões correspondem a modelos nos quais adicionam-se regras de funcionamento com a finalidade de enriquecer a capacidade de representação do modelo inicial, possibilitando uma quantidade maior de aplicações a serem tratadas. Três subclasses podem ser consideradas:

1. as que correspondem aos modelos que têm o poder de representação de máquinas de Turing: RdP com arcos inibidores e RdP prioritárias;

2. as que correspondem a extensões que permitem a modelagem de sistemas contínuos ou híbridos: RdP contínua e RdP híbrida;
3. as correspondentes às redes de Petri não autônomas, que descrevem o funcionamento de sistemas cuja evolução leva em consideração eventos externos e/ou tempo: RdP sincronizada, RdP temporizada, RdP interpretada e RdP estocástica. Nesta subclasse nem todas as propriedades de uma RdP Ordinária são mantidas.

### 3.1.1 Descrição dos tipos de redes de Petri

Nesta seção os principais modelos em RdP encontrados na literatura serão descritos brevemente, no intuito de mostrar como as RdP são usadas para modelar os diversos tipos de sistemas a eventos discretos, de forma que ao final seja possível estabelecer quais desses modelos são os mais adequados para o nosso estudo. Informações mais detalhadas dos modelos podem ser encontradas em David & Alla (1994).

**Ordinárias –** As Redes de Petri Ordinárias, também chamadas de primitivas (DAVID & ALLA, 1994), possuem baixo poder de modelagem por representarem apenas relações de causa e efeito entre os eventos e as condições. A sua utilização é restrita portanto a diversos tipos de sistemas pertencentes a classe de sistemas dinâmicos de eventos discretos, onde sincronização externa e o tempo não intervêem. Desde que concorrência, sincronização e compartilhamento de recursos possam ser achados na especificação de tais sistemas, as Redes de Petri são uma ferramenta muito apropriada para sua modelagem (MURATA, 1981; BERTHELOT, 1982). Um dos campos de aplicação mais freqüentes é o de sistemas de manufatura (VALETTE & SILVA, 1990), pois a concorrência (duas máquinas trabalhando independentemente), a sincronização (uma máquina está livre esperando uma peça ficar pronta para ser processada pela mesma) e o compartilhamento de recursos (um robô é requerido para manusear peças por duas máquinas, mas não pode servir ambas ao mesmo tempo), são características usuais de tais sistemas.

As subclasses mais conhecidas desta classe são: as máquinas de estados, os grafos de eventos, as redes de Petri de escolha livre e as redes de Petri simples. Cada uma dessas subclasses possui determinadas propriedades e características importantes para o desempenho dos sistemas modelados por elas. Devido a esta importância estas redes serão estudadas na seção 3.4.

**Abreviações –** As abreviações tem a finalidade de melhorar o entendimento do grafismo e correspondem a representações simplificadas das RdP. Qualquer RdP abreviada deve sempre corresponder a alguma RdP ordinária e por causa desta exigência todas as propriedades da RdP ordinária são mantidas na RdP abreviada. É necessário observar que as abreviações não aumentam o poder de modelagem, mas sim sua conveniência. A seguir serão descritas sucintamente algumas dessas abreviações:

1. RdP Generalizada - É uma RdP onde pesos (valores inteiros estritamente positivos) são associados aos arcos. Quando um arco $p_i \rightarrow t_j$ tem peso $w$, significa que a transição

$t_j$ estará habilitada somente se o lugar $p_i$ contiver pelo menos $w$ fichas. Quando esta transição for disparada ela irá retirar $w$ fichas de $p_i$. Se um arco $t_j \rightarrow p_i$ tem peso $v$, o disparo de $t_j$ irá depositar $v$ fichas no lugar $p_i$.

2. RdP com Capacidade Finita - É uma RdP onde a cada lugar é associado uma certa capacidade de fichas. O disparo de uma transição de entrada de um lugar capacitado $p$ somente é possível se, o disparo desta transição não resultar em uma quantidade de fichas que exceda a capacidade de $p$.
3. RdP Coloridas - Em uma RdP Colorida existe mais do que um tipo de ficha e as fichas são *coloridas* (o que equivale a terem rótulos associados). A cada lugar se associa o conjunto de cores das fichas que podem pertencer a este lugar. A cada transição se associa um conjunto de cores que corresponde às diferentes maneiras de disparar uma transição. Nos casos mais simples, quando todos os processos possuem rigorosamente a mesma estrutura e são independentes uns dos outros, as cores das transições são diretamente associadas aos processos, e o conjunto de cores dos lugares e das transições são idênticos (CARDOSO & VALETTE, 1997).

**Extensões –** As extensões correspondem a modelos nos quais adicionam-se regras de funcionamento para enriquecer a modelagem inicial, por causa deste acréscimo nem todas as propriedades das redes de Petri ordinárias são mantidas. A seguir serão mostradas as principais extensões:

1. RdP com Arcos Inibidores - Um arco inibidor é um arco dirigido que une um lugar a uma transição com seu extremo final sendo marcado por um círculo pequeno. Este lugar deve ter no mínimo duas transições de saída $t_1$ e $t_2$, sendo que uma delas (por exemplo $t_1$) está ligada a $p$ através do arco inibidor, significando que a transição $t_1$ só poderá disparar se $p$ não contiver nenhuma ficha. O disparo de $t_1$ consiste em tomar uma ficha de cada um de seus lugares de entrada, com exceção de $p$, depositando uma ficha em cada um de seus lugares de saída. Se $p$ contiver uma ficha, $t_1$ só poderá disparar após o disparo de $t_2$ e até que $p$ não contenha mais fichas.

   De acordo com Murata (1989), as expressões "teste-zero"e "RdPs estendidas"são freqüentemente usadas na literatura para se referir aos arcos inibidores.
2. Redes de Petri Contínuas - A característica principal destas redes de Petri é que as marcações dos lugares são números reais positivos e não mais números inteiros. O disparo de uma transição é um fluxo contínuo. Estas redes representam sistemas que não podem ser modelados por RdP Ordinárias, por exemplo, quando o número de marcações de uma RdP Ordinária torna-se muito grande. Este modelo é autônomo, pois o tempo não é considerado na modelagem. Geralmente, os lugares são representados por dois círculos e as transições por retângulos para diferenciá-las dos lugares e das transições nas RdPs Ordinárias.
3. Rede de Petri Híbrida - Este é um novo modelo apresentado pela primeira vez por Le Bail em 1991 (DAVID & ALLA, 1994). Esta rede é formada tanto por lugares e transições

discretas quanto lugares e transições contínuas.

As RdP autônomas são por definição aquelas em que uma transição pode ser disparada a qualquer tempo assim que estiver habilitada, mas não é possível saber *a priori* quando ela será disparada.

As extensões mostradas a seguir são também conhecidas na literatura como RdP não-autônomas. Elas permitem descrever, o que acontece no sistema modelado, sob a influência de eventos externos. Estas RdP permitem que sistemas sejam modelados quando os disparos das transições são sincronizados por eventos externos, e/ou cujas evoluções são dependentes do tempo.

A seguir será feito uma breve descrição das principais RdP não-autônomas:

**Redes de Petri dependentes de eventos externos**

4. Redes de Petri Sincronizadas - Numa RdP Sincronizada, algum evento externo à rede é associado a cada transição, e o disparo desta transição acontecerá:
   *se* a transição estiver habilitada e
   *quando* o evento associado ocorrer.

**Redes de Petri dependentes do tempo**

A definição de redes de Petri dependentes do tempo compõe-se por três especificações:

(a) estrutura topológica - geralmente é a mesma da Rede de Petri ordinária;
(b) rotulagem da estrutura - feita através da atribuição de valores numéricos a um ou mais elementos da rede (transições, lugares e arcos);
(c) regras de disparo - são dependentes do tipo de rotulagem das variáveis de tempo e controlam o processo de movimento das fichas na rede.

5. Rede de Petri T-Temporizada - A rede de Petri t-temporizada associa a cada transição da rede um único parâmetro temporal (sua duração de disparo). Um tempo é associado a cada transição. No momento em que uma transição torna-se habilitada ela dispara, seu disparo absorve as fichas correspondentes de cada um dos seus lugares de entrada, as fichas devem permanecer na transição durante o tempo da execução deste disparo e apenas quando a duração do disparo terminar as fichas serão depositadas em cada lugar de saída da transição.
6. Rede de Petri P-Temporizada - Uma rede de Petri p-temporizada associa a cada lugar um tempo. Quando uma ficha é depositada no lugar, a mesma deverá permanecer neste lugar no mínimo o tempo associado a este lugar (esta ficha é dita ser indisponível por este tempo). Quando o tempo decorreu, as fichas então tornam-se disponíveis. Somente fichas disponíveis podem ser consideradas para habilitar condições. Sifakis(1979) demonstrou a equivalência entre as RdP p-temporizadas e RdP t-temporizadas, por isso as redes de Petri p- e/ou t-temporizadas, são chamadas comumente de Redes de Petri Temporizadas.
7. Rede de Petri Temporal - Esta RdP consiste na atribuição de um intervalo de tempo de disparo $[T_{min}, T_{max}]$ para cada $t_i \in T$. Neste caso:

$T_{min}$ = Tempo mínimo de espera para ti poder disparar após habilitar-se.
$T_{max}$ = Tempo máximo em que ti pode disparar após habilitado.

Assim, se $t_i \in T$ for habilitado no instante $\theta$, a transição $t_i$ só poderá disparar no intervalo $[T_{min} + \theta, T_{max} + \theta]$. Em outras palavras, uma transição deve permanecer sensibilizada durante a espera mínima $T_{min}$ antes de poder ser disparada, e não pode disparar além da espera máxima $T_{max}$. O disparo de uma transição tem duração nula, hipótese essencial ao funcionamento deste modelo de RdP. A análise de uma RdP Temporal é bastante difícil devido a grande quantidade de estados.

8. Rede de Petri Estocástica - Em RdP Temporizadas, uma duração fixa (ou uma janela de tempo fixa), é associada a cada lugar ou a cada transição da rede. Porém existem casos, onde a RdP não pode ser modelada com durações constantes, como por exemplo, o tempo de funcionamento real entre 2 quebras de máquina. Esta duração pode ser modelada por uma variável aleatória. Neste caso, pode-se associar um tempo aleatório ao disparo de cada transição e geralmente com o tempo sendo distribuído segundo uma lei exponencial. Desta forma a marcação da RdP Estocástica é um processo markoviano homogêneo, assim uma cadeia de markov homogênea poderá ser associada a cada RdP estocástica (MOLLOY, 1982; MOLLOY, 1985).

**Redes de Petri dependentes de eventos externos e do tempo**

- Redes de Petri Interpretadas - são RdP que recebem algum tipo de interpretação. Peterson (1981) define as redes de Petri interpretadas, como sendo as redes às quais foram associadas alguma interpretação, ou significado, aos seus lugares e transições, dando um significado real ao que se pretende modelar. As redes de Petri não-interpretadas são as que não foram associadas qualquer interpretação aos seus componentes, sendo uma representação totalmente abstrata (PETERSON, 1981).

  Mas a definição mais utilizada no estudo de sistemas a eventos discretos é: uma rede de Petri interpretada (RdPI) é uma rede de Petri sincronizada, p-temporizada, com variáveis associadas às suas transições, representando condições e ações existentes no sistema. Uma RdPI é uma extensão de RdP que possibilita a representação de sinais de saída gerados quando uma marcação é alcançada, e de sinais de entrada associadas às transições que são controláveis (DAVID & ALLA, 1994).

## 3.2 Propriedades de Redes de Petri

A importância da modelagem em Rede de Petri de sistemas reais reside na possibilidade do mesmo modelo proporcionar tanto a análise das propriedades comportamentais e estruturais do sistema modelado e sua avaliação de desempenho, quanto a construção sistemática de simuladores e controladores desse sistema (MURATA, 1989). A análise do modelo leva a uma compreensão melhor sobre o comportamento e as propriedades deste sistema. Dois tipos de propriedades podem ser estudados com redes de Petri (MURATA, 1989):

- Propriedades que dependem do estado inicial, ou da marcação, denominadas propriedades

comportamentais ou dependentes de marcação, sendo as mais importantes a alcançabilidade, a limitação, a vivacidade, a reversibilidade, a cobertura, a persistência, a distância sincrônica e a justiça.

- Propriedades independentes da marcação inicial e dependentes da topologia da rede, denominadas propriedades estruturais, sendo as mais importantes a vivacidade estrutural, a controlabilidade, a limitação estrutural, a conservabilidade, a repetitividade e a consistência.

Uma extensiva descrição de todas estas propriedades pode ser encontradas em Murata (1989).

Baseando-se no formalismo de redes de Petri apresentado na seção 2.3.1 e dada uma RdP definida por $N =< P, T, I, O, M_0 >$, a seguir serão definidas as propriedades mais relevantes para o caso tratado neste trabalho.

### 3.2.1 Propriedades Comportamentais

As propriedades comportamentais dependem da marcação inicial e estão ligadas à evolução da rede.

- Um lugar $p \in P$ é k-limitado se $\forall m \in R(N, m_0), \ \exists k \in \mathbb{N} \mid m(p) \leqslant k$.
- N é k-limitada se p é k-limitado, $\forall p \in N$.
  N é segura se se ela é 1-limitada.
- N é viva se a partir de qualquer estado alcançável $m \in R(N, m_0)$ e $\forall t \in T$, existe uma seqüência de disparos $\sigma$ que habilita t.
- Uma marcação $m'$ é um estado–base (*homestate*) se para cada marcação $m \in R(N, m_0)$, $m'$ é alcançável a partir de m.
- N é reversível se $m_0$ é um estado–base.

### 3.2.2 Propriedades Estruturais

As propriedades estruturais estão ligadas a topologia da rede e são independentes da marcação inicial $m_0$. Dada uma rede de Petri N, estas propriedades podem ser caracterizadas em termos da matriz de incidência C e suas equações e inequações associadas.

Baseando-se no formalismo de redes de Petri apresentado na seção 2.3.1 e dada uma rede de Petri $N =< P, T, I, O, M_0 >$, com r transições e s lugares, sua matriz de incidência $C = \left[c_{ij}\right]$ é uma matriz $(s \times r)$ definida por:

$$C(p_i, t_j) = \left[c_{p_i,t_j}\right] = O\left(p_i, t_j\right) - I\left(p_i, t_j\right) \tag{3.1}$$

A equação fundamental de redes de Petri é estabelecida em função da matriz de incidência:

$$m' = m_0 + C.y \tag{3.2}$$

onde $m_0$ é sua marcação inicial da rede, $m'$ é a marcação obtida a partir de $m_0$ pela sequência de disparos $\sigma$ e $y$ é o vetor característico associado à sequência $\sigma$. O vetor $y$ é determinado da seguinte maneira:

Seja $\sigma$ uma seqüência de d-transições caracterizada por $j(i) : \{1, 2, \ldots, d\} \rightarrow T$ que associa a transição $t_j$ ao i-ésimo disparo. Seja $u_j = [0, 0, \ldots, 1, \ldots, 0]^t$ um vetor cujo único componente não nulo é o j-ésimo elemento. Finalmente:

$$y = \sum_{i=1}^{d} u_{j(i)}.$$

.

A equação 3.2 pode ser reescrita da seguinte forma:

$$C.y = m' - m_0 = \Delta m \tag{3.3}$$

Com base na equação fundamental, serão definidas a seguir, as propriedades mais relevantes para este trabalho:

- Vivacidade Estrutural – Uma rede de Petri N é dita ser *estruturalmente viva* se existe uma marcação inicial para a qual N é viva.

  Segundo DiCesare *et. al.* (1993), uma condição necessária para que um transição t seja viva em uma rede N com marcação inicial $m_0$, é a de que eventualmente poder disparar infinitamente, isto é, é a de existir uma seqüência repetitiva de disparos $\sigma_r$ incluindo o disparo de t.

- Controlabilidade – Uma rede de Petri N é dita ser *completamente controlável* se uma marcação qualquer é alcançada a partir de qualquer outra marcação.

  Para esta propriedade Murata (1989) propôs o seguinte teorema:

  **Teorema 3.1** *Se uma rede de Petri* N *com* s *lugares é completamente controlável então o posto (rank) de sua matriz de incidência* C *será:*

  $$\text{Posto}(C) = s. \tag{3.4}$$

  **Demonstração:** *Se uma rede de Petri é completamente controlável, então a equação 3.3 deve ter uma solução* S *para qualquer* $\Delta m$*. Pela resolução de sistemas lineares sabe-se que para qualquer* $\Delta m$*:*

  $$\text{Posto}(C) = \text{Posto}\left[C \vdots \Delta m\right];$$

  *logo,* $C_{s\times r}$ *deve ter posto igual a* s*.*

  A equação 3.4 é apenas uma condição necessária para a completa controlabilidade de redes de Petri (MURATA, 1989).

- Limitação Estrutural – Uma rede de Petri N é denominada *estruturalmente limitada* se ela é limitada para qualquer marcação inicial finita $m_0$.

  Para esta propriedade Murata (1989) propôs o seguinte teorema:

**Teorema 3.2** *Uma rede de Petri* $N$ *é limitada estruturalmente se e somente se existe um vetor* $x$ *de inteiros positivos tal que*

$$x^t \cdot C \leqslant 0, , \quad x \neq 0. \tag{3.5}$$

A demonstração deste teorema pode ser encontrada em Murata (1989).

Um lugar $p$ em uma rede de Petri é *estruturalmente ilimitado* se existe uma marcação $m_0$ e uma seqüência de disparos $\sigma$ a partir de $m_0$ tal que $p$ seja ilimitado (MURATA, 1989).

- Conservação Estrita – Uma rede de Petri marcada $N$ é denominada *estritamente conservativa* se:

$$\sum_{p_i \in P} m(p_i) = \sum_{p_i \in P} m_0(p_i), \qquad \forall m \in R(N, m_{0)}.$$

Conservação estrita é uma propriedade muito restritiva, pois ela indica que existe exatamente o mesmo número de fichas em todas as marcações alcançáveis de uma rede de Petri. Do ponto de vista estrutural, isto só ocorrerá se o número de arcos de entradas de cada transição for igual ao número de arcos de saídas, $\left|I(t_j)\right| = \left|O(t_j)\right|$. Se este não for o caso, o disparo da transição $t_j$ poderá mudar o número de fichas na rede (PETERSON, 1981).

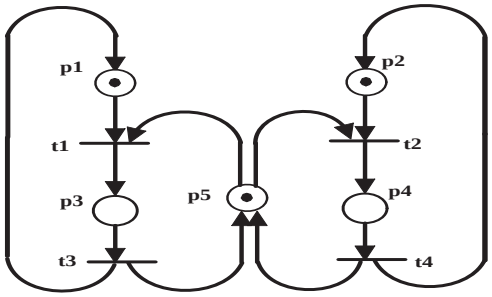



**Figura 3.1**: Rede de Petri não estritamente conservativa

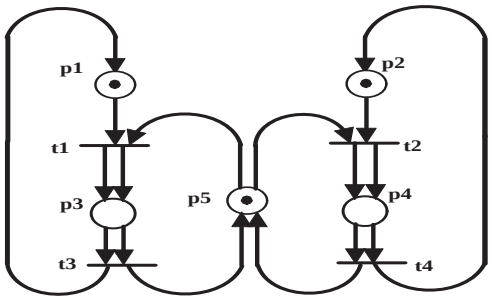



**Figura 3.2**: Rede de Petri conservativa e equivalente a rede da fig. 3.1.

A figura 3.1 ilustra melhor esta idéia. A RdP da figura 3.1 não é estritamente conservativa já que o disparo ou da transição $t_1$ ou da transição $t_2$ decrescerá o número de fichas de uma unidade, enquanto o disparo ou da transição $t_3$ ou da transição $t_4$ adicionará uma ficha na marcação. No entanto, pode-se converter a Rede de Petri da figura 3.1 para a da figura 3.2, que é estritamente conservativa (Peterson, 1981).

Em sistemas reais, recursos são frequentemente combinados para que uma tarefa seja executada e depois são separados quando esta tarefa termina. Uma forma de superar este problema é associar pesos aos lugares, possibilitando que a soma dos pesos das fichas na rede seja constante. Isto resulta em uma definição mais ampla para a conservação (WANG, 1998):

**Definição 3.1** *Uma rede de Petri marcada* $N$ *é dita ser (parcialmente) conservativa com respeito a um vetor de pesos* $w = (w_1, w_2, \ldots, w_n)$*, se* $\forall m \in R(N, m_0)$*:*

$$\sum_{i=1}^{n} w(i)m(p_i) = \sum_{i=1}^{n} w(i)m_0(p_i).$$

onde $n$ é o número de lugares e $w_i > 0$ ($w_i \geqslant 0$) e de tal forma que $\sum_{i=1}^{n} w(i) \neq 0$.

**Teorema 3.3** *Uma rede de Petri* $N$ *viva é conservativa (parcialmente) se e somente se existe um vetor* $x$ *de inteiros positivos (não-negativos) tal que*

$$x^t \cdot C = 0, \quad x \neq 0. \tag{3.6}$$

A demonstração deste teorema pode ser encontrada em Memmi & Roucairol (1980), Lautenbach (1986) e Murata (1989).

Note que conservação é um caso particular da limitação estrutural.

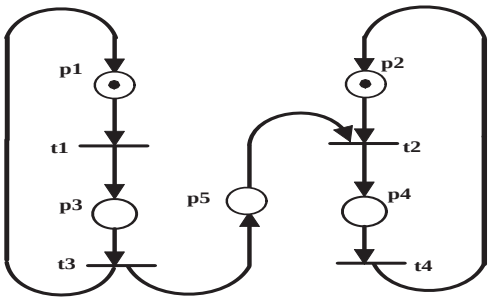



**Figura 3.3**: Rede de Petri não conservativa

A RdP da figura 3.1 é conservativa com relação a $(1, 1, 2, 2, 1)$ e a RdP da figura 3.3 não é conservativa e também não é limitada (PETERSON, 1981).

Quando uma rede é estritamente conservativa, a soma de todos as fichas permanece imutável para todas as marcações alcançáveis. Tais redes de Petri geram modelos para sistemas com um número constante de carretos (*carts*), móveis (fixture), esteiras, ou *jobs*, como por exemplo, um sistema de fila fechado ou um sistema de transporte. Conservação implica em limitação (ZHOU *et al.*, 1993), pois se o número de fichas é sempre constante para todas as marcações alcançáveis, então a quantidade de fichas em cada lugar é limitada a esta constante.

- Repetitividade

  Dada uma rede de Petri N com $s$ lugares e sendo C sua matriz de incidência, define-se repetividade da seguinte forma (MURATA, 1989):

  **Definição 3.2** *Uma rede de Petri* N *com* $s$ *lugares é dita ser (parcialmente) repetitiva se existe uma marcação* $m_0$ *e uma seqüência de disparos* $\sigma$ *partindo de* $m_0$*, tal que (algumas) todas transições ocorram com uma freqüência infinita em* $\sigma$*.*

  **Teorema 3.4** *Uma rede de Petri* N *com* $s$ *lugares é (parcialmente) repetitiva se e somente se existe um vetor* $y$ *de inteiros positivos (não-negativos) tal que*

  $$C \cdot y \geqslant 0, \quad y \neq 0. \tag{3.7}$$

  A demonstração deste teorema pode ser encontrada em Murata (1989).

- Consistência

  A consistência (ou repetitividade estacionária) de uma rede de Petri N, com $s$ lugares e com C representando a matriz de incidência de N, pode ser definida da seguinte forma:

  **Definição 3.3** *Uma rede de Petri* N *com* $s$ *lugares é dita ser (parcialmente) consistente se existe uma marcação* $m_0$ *e uma seqüência de disparos* $\sigma$ *partindo de* $m_0$ *e voltando a* $m_0$*, tal que (algumas) todas transições da rede ocorram pelo menos um vez em* $\sigma$*.*

  **Teorema 3.5** *Uma rede de Petri* N *com* $s$ *lugares é (parcialmente) consistente se e somente se existe um vetor* $y$ *de inteiros positivos (não-negativos) tal que*

  $$C \cdot y = 0, \quad y \neq 0. \tag{3.8}$$

  Caso contrário, a rede de Petri é inconsistente.

  A demonstração deste teorema pode ser encontrada em Memmi & Roucairol (1980), Lautenbach (1986) e Murata (1989).

  Um sistema é consistente (inconsistente) se sua modelagem em Rede de Petri é consistente (inconsistente).

  Note que consistência é um caso particular de repetitividade, por isso pode ser denominada de repetitividade estacionária.

  A figura 3.4(a) ilustra um sistema inconsistente e a figura 3.4(b) um sistema consistente. Na figura 3.4(a) não existe nenhuma atribuição de inteiros para suas transições que satisfaça a condição de consistência. Isto pode ser verificado atribuindo-se três variáveis inteiras $n_1$, $n_2$

e $n_3$ para as transições $t_1$, $t_2$ e $t_3$, respectivamente, gerando uma contradição na tentativa de solucionar simultaneamente as equações fornecidas pela condição de consistência:

$$\text{para o lugar } p_1: \quad n_1 + n_2 = n_3; \tag{3.9}$$

$$\text{para o lugar } p_2: \quad n_1 = n_3; \tag{3.10}$$

$$\text{para o lugar } p_3: \quad n_2 = n_3 \tag{3.11}$$

$$\text{equação (3.10)} + \text{equação (3.11)}: \quad n_1 + n_2 = 2n_3. \tag{3.12}$$

Portanto a equação 3.12 contradiz a equação 3.9.

A figura 3.4(b) ilustra uma rede de Petri consistente. Se a cada transição for atribuído um peso inteiro de valor 1, a condição de consistência é satisfeita.

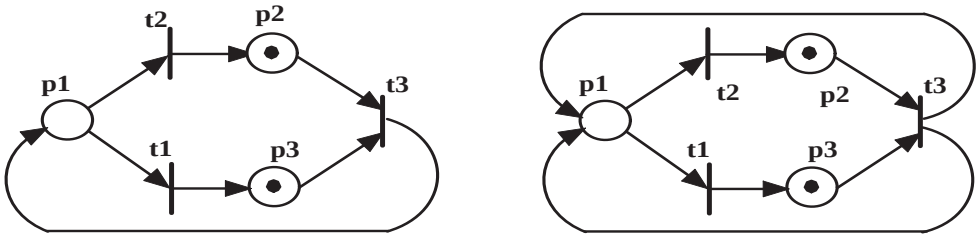



**Figura 3.4**: (a)Rede de Petri inconsistente. (b)Rede de Petri consistente.

A implicação prática desta classificação do sistema é que o peso (nº inteiro) atribuído à transição é o número relativo de execuções da transição em um ciclo. Se um sistema é vivo e consistente, o sistema voltará ao seu estado inicial depois de cada ciclo e então repete-se. Se um sistema é inconsistente, ou ele produz um número infinito de fichas (isto é, ele precisa de recursos infinitos) ou consome fichas e consequentemente leva a uma parada. Os sistemas do mundo real que funcionam continuamente com quantidade finita de recursos caem na classe de sistemas consistentes (WANG, 1998).

Segundo DiCesare *et. al.*(1993), considerando-se que a repetitividade estrutural é uma condição necessária para a vivacidade estrutural, é possível relacionar-se as propriedades estruturais, chegando a um resultado clássico na literatura (MEMMI & ROUCAIROL, 1980).

Este resultado é muito importante para o desenvolvimento deste trabalho e é mostrado a seguir:

**Propriedade 3.1** *Se uma rede de Petri* $N =< P, T, I, O >$ *é estruturalmente viva e estruturalmente limitada então:*

$$\exists y > 0, \quad \textit{tal que} \quad C \cdot y = 0, \quad \textit{isto é,}\ N\ \textit{é consistente.}$$

$$\exists x > 0, \quad \textit{tal que} \quad x^t \cdot C = 0, \quad \textit{isto é,}\ N\ \textit{é conservativa.}$$

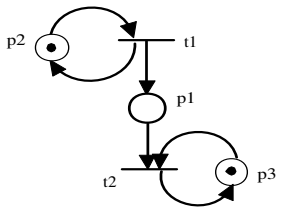



**Figura 3.5**: Rede de Petri Consistente.

A hipótese de que a rede é limitada e viva é uma condição suficiente mas não necessária para a ocorrência da consistência e da conservação. É possível a consistência em redes não vivas, da mesma forma que em redes não limitadas. Isto pode ser visto através do exemplo na figura 3.5.

A rede na fig.3.5 não é limitada. Entretanto, a rede é consistente. De fato, se $t_1$ for disparado e logo em seguida $t_2$ e mantendo-se esta seqüência, o funcionamento será repetitivo estacionário.

As redes de Petri da figura 3.6 são consistentes e conservativas, mas não são vivas.

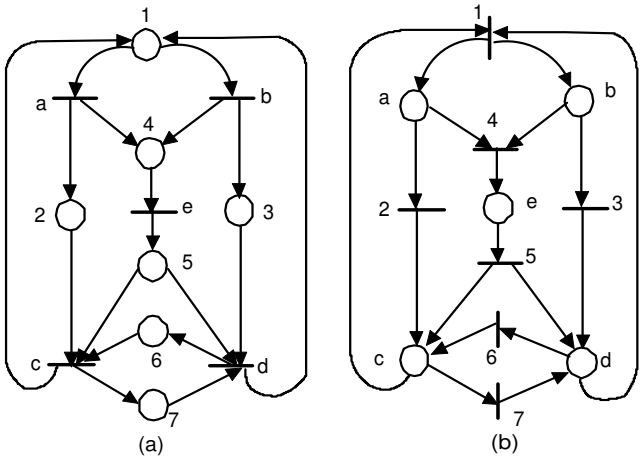



**Figura 3.6**: Redes de Petri Consistentes e Conservativas (DiCesare *et. al.*, 1993).

De acordo com DiCesare *et. al.*(1993)[1], é possível aperfeiçoar o resultado dado pela propriedade 3.1 incluindo a condição do *posto* da matriz de incidência C da rede de Petri N. Para incluir esta condição é necessário antes definir a quantidade $\delta$ da seguinte forma (DICESARE *et. al.*, 1993):

Seja $t_i$ e $t_j$ duas transições da rede de Petri N. Dizemos que $t_i$ e $t_j$ tem uma **relação de conflito em igualdade** se $I(p, t_i) = I(p, t_j) \neq 0$. Esta é uma relação de equivalência que leva ao particionamento do conjunto de transições em classes de equivalência. Seja $D_k$ uma classe de equivalência. A quantidade $\delta$ será definida da seguinte forma:

$$\begin{aligned} \delta_k &= |D_k| - 1, \\ \delta &= \sum_k \delta_k. \end{aligned}$$

[1]DiCesare *et. al.*(1993), pág. 42

Agora já é possível aperfeiçoar o resultado dado pela propriedade 3.1:

**Propriedade 3.2** *Se uma rede de Petri* $N =< P, T, I, O >$ *é estruturalmente viva e estruturalmente limitada então:*

$$\begin{aligned} &\exists y > 0, \quad \textit{tal que} \quad C \cdot y = 0, \quad \textit{isto é,}\ N\ \textit{é consistente.} \\ &\exists x > 0, \quad \textit{tal que} \quad x^t \cdot C = 0, \quad \textit{isto é,}\ N\ \textit{é conservativa.} \\ &posto(C) \quad \leqslant \quad |T| - \delta - 1, \end{aligned} \tag{3.13}$$

Segundo DiCesare *et. al.*(1993), a adição da condição do posto de C permite estabelecer que a rede da figura 3.6(a) é estruturalmente não viva, pois $\delta = 1$ e $posto(C) = 4, |T| - \delta - 1 = 5 - 1 - 1$, o que leva a: $posto(C) > |T| - \delta - 1$. No entanto, nada pode ser dito sobre a vivacidade estrutural da rede da figura 3.6(b), pois $\delta = 2$ e $posto(C) = 4, |T| - \delta - 1 = 7 - 2 - 1$, o que leva a: $posto(C) = |T| - \delta - 1$.

Ainda segundo DiCesare *et. al.*(1993), a propriedade 3.2 é puramente estrutural, pois não leva em consideração a marcação inicial da rede.

### 3.2.3 Análise das propriedades

- **Análise por Enumeração**

  Segundo Desrochers & Al-Jaar (1995), os problemas de alcançabilidade e de cobertura são questões de análise de redes de Petri muito básicos e são úteis para o estabelecimento de propriedades tais como limitabilidade e reversibilidade.

  O problema de alcançabilidade pode ser definido da seguinte forma: *Dada uma rede de Petri N com duas marcações diferentes* $m$ *e* $m'$*, é possível determinar se* $m'$ *é alcançável a partir de* $m$ *(isto é,* $m' \in R(N, m)$*)?*

  Dado que um estado $m'$ é dito ser coberto por um estado $m''$ se, todos os componentes de $m''$ são maiores ou iguais aos correspondentes componentes de $m'$, o problema de cobertura pode ser definido da seguinte forma: *Dada uma rede de Petri N com uma marcação inicial* $m_0$ *e uma marcação* $m'$*, existe uma marcação alcançável* $m'' \in R(m_0)$*, tal que* $m''$ *cobre m' (isto é,* $m'' \geqslant m'$*)?*

  A verificação das propriedades de limitabilidade e reversibilidade pode ser feita pela construção da árvore de alcançabilidade que é a representação de todas as marcações que a RdP pode alcançar a partir da marcação inicial (MURATA, 1989). Cada nó da árvore corresponde a uma marcação alcançável e cada arco corresponde ao disparo de uma transição, que faz passar de uma marcação a outra (KHANSA, 1997). Porém é possível que esta árvore cresça sem que seja possível determinar seu fim.

  A construção da árvore de cobertura permite decidir se uma RdP é limitada. A árvore de cobertura é definida de modo similar à árvore de alcançabilidade exceto pelo fato de que as marcações que crescem indefinidamente são substituídas por uma única marcação que

contém o símbolo $\omega$ (número finito, indefinidamente grande). Desse modo, a árvore de cobertura é finita, com a presença do símbolo $\omega$ na árvore indicando que qualquer estado obtido pela substituição deste símbolo por algum número inteiro é coberto por algum estado alcançável.

A RdP é limitada se e somente o símbolo $\omega$ não aparece na árvore de cobertura. Através desta árvore, é possível verificar as outras propriedades. Entretanto, para uma RdP ilimitada, construir a árvore de cobertura não é suficiente para resolver o problema de alcançabilidade e o de vivacidade. Estes dois problemas são decidíveis[2], mas com algoritmos muito mais complicados (KHANSA, 1997). Este método é válido apenas para a rede marcada, e a cada marcação inicial diferente corresponde uma nova árvore. Esta abordagem de análise é geral, pois ela é aplicável a todas as classes de rede. Sua limitação reside na explosão combinatorial do número de estados (KHANSA, 1997). A abordagem baseada no estudo de RdP através de sua árvore de cobertura constitui a **Análise Enumerativa** ou por Enumeração.

- **Análise por Invariantes**

  Os problemas de conservação e de consistência mostrados na seção 3.2.2, podem ser tratados através da Análise por Invariantes. Esta análise consiste em determinar algumas das propriedades estruturais a partir da matriz de incidência da rede de Petri. Conforme o teorema 3.2 a existência de uma solução não nula para a equação $x_t.C = 0$ é condição necessária e suficiente para a conservatividade da rede de Petri.

  - **P-invariante** – um p-invariante (lugar invariante) é um vetor de inteiros positivos $x$ de dimensão $s$ que satisfaz:

    $$x^t.C = 0 \tag{3.14}$$

    Sendo $x$ um p-invariante, $m_0$ a marcação inicial de uma RdP e $m$ uma marcação alcançável a partir de $m_0$ tem-se:

    $$x^t.m = x^t.m_0 \tag{3.15}$$

    Este resultado é uma conseqüência da equação fundamental.

    Deve-se observar que as soluções da equação 3.15 estão no núcleo ($kernel$) da matriz $C_t$, logo o número de soluções linearmente independentes é igual à dimensão deste espaço.

  O conjunto de lugares correspondentes a elementos não nulos em um p-invariante $x \geqslant 0$ é denominado *suporte deste invariante* ou *componente conservativo* e é denotado por $\|x\|$. Um suporte é *minimal* se não existe outro invariante $x_1$ tal que $x_1(p) \leqslant x(p)$ para todo $p$. Dado um suporte minimal de um invariante, existe um único invariante minimal correspondendo ao suporte minimal. Tal invariante é denominado *suporte–minimal do invariante*.

[2]Para mostrar a decidibilidade de um problema de rede de Petri, é necessário reduzir este problema a um problema com uma solução conhecida. Para mostrar indecidibilidade, deve-se reduzir este problema a outro conhecido como sendo indecidível (PETERSON, 1980).

A existência de p-invariantes implica na conservação da rede conforme o teorema 3.3.

- **T-invariante**– um T-invariante é um vetor de inteiros positivos $y$ de dimensão $r$ que satisfaz:

$$C.y = 0 \tag{3.16}$$

  Seja $\sigma$ uma seqüência de disparo e $S$ seu vetor característico. Sendo $S = y$ um T-invariante, e sendo $m_0$ uma marcação inicial de $N$ e $m$ uma marcação alcançável a partir de $m_0$ tem-se:

$$m = m_0 \tag{3.17}$$

  Este resultado é imediato a partir da equação fundamental e mostra que $\sigma$ é uma seqüência cíclica, isto é, o disparo da seqüência $\sigma$ faz o sistema retornar à marcação inicial da rede (a rede é reiniciável).

O conjunto de transições correspondentes a elementos não nulos em um t-invariante $y \geqslant 0$ é denominado *suporte deste invariante* ou *componente repetitivo estacionário* e é denotado por $\|y\|$. Um suporte é *minimal* se não existe outro invariante $y_1$ tal que $y_1(p) \leqslant y(p)$ para todo $p$.

A existência de t-invariantes implica na consistência da rede conforme o teorema 3.5.

Segundo Desrochers & Al-Jaar (1995) um resultado importante da t-invariância é dado por:

**Propriedade 3.3** *Dada a rede de Petri* $N =< P, T, I, O >$ *e sua matriz de incidência* $C$, *se a equação* $Cy = 0$ *só admite a solução trivial, então* $N$ *não é reversível.*

Esta é uma condição suficiente, mas não é necessária para a irreversibilidade. Donde, a solução não trivial garante apenas a reversibilidade parcial.

O conjunto de todos os possíveis suportes minimais de invariantes serve como um gerador de invariantes, isto é, qualquer invariante pode ser escrito como uma combinação linear dos suportes-minimais de invariantes (WANG, 1998).

Este conjunto de suporte minimais de invariantes proporciona uma visão decomposta da estrutura da rede de Petri estudada e esta decomposição torna possível implementar a decomposição das redes de Petri em processos (ver seção 2.4), com cada suporte minimal dos p-invariantes sendo tratado como um processo e com cada suporte minimal dos t-invariantes sendo usado para verificar a repetitividade da rede.

Estes resultados são interessantes no contexto deste trabalho, pois tornam possível a transposição (ou tradução) da modelagem em redes de Petri para a modelagem em programação linear inteira mista, da seguinte forma:

– na seção 3.5, serão estudados os invariantes característicos dos processos simples, com sincronismo, com alternância e com compartilhamento (definidos na seção 2.4);

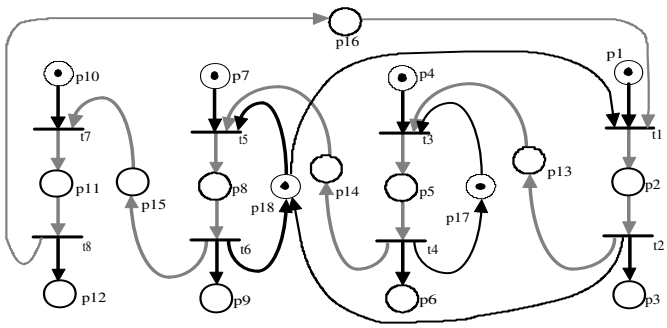



**Figura 3.7**: Problema de Escalonamento com Precedências Conflitantes

– no cap. 5 será proposto um algoritmo para encontrar o conjunto dos suportes minimais dos invariantes, decompôr a rede de Petri em componentes conservativos e modelar cada componente de acordo com a formulação proposta no cap. 4.

De acordo com Aalst (1995), é possível encontrar precedências conflitantes e precedências redundantes usando-se a análise da p-invariância.

A precedência conflitante acontece quando um p-invariante não contém fichas na marcação inicial, sendo portanto uma seqüência não realizável. Para ilustrar este resultado considere o problema de escalonamento mostrado na figura 3.7. Seja o componente conservativo gerado pelo conjunto (Aalst, 1995):

$$\{p_2, p_{13}, p_5, p_{14}, p_8, p_{15}, p_{11}, p_{16}\}$$

O seu p-invariante é:

$$p_2 + p_{13} + p_5 + p_{14} + p_8 + p_{15} + p_{11} + p_{16} = 0$$

Esta invariante mostra que existem 4 restrições de precedência em conflito, representadas pelos lugares $p_{13}, p_{14}, p_{15}$ *e* $p_{16}$ e seus arcos (ver figura 3.7). Uma forma de tornar esta seqüência realizável é remover um destes lugares e seus respectivos arcos.

As restrições de precedência redudante ocorrem quando os lugares que representam estas restrições são redundantes, como conseqüência, a retirada de um destes lugares não afeta o comportamento da rede. As restrições de precedências redundantes também podem ser encontradas usando-se os p-invariantes. Por exemplo, na figura 3.7 se for mudada a direção dos arcos de entrada e de saída do lugar $p_{16}$ (que simbolizam uma restrição de precedência do problema de escalonamento), será obtida uma rede de Petri sem precedências conflitantes (mostrado pela figura 3.8), gerando o seguinte P-invariante:

$$p_2 + p_{13} + p_5 + p_{14} + p_8 + p_{15} + p_{11} - p_{16} = 0$$

Este invariante mostra que $p_{16}$ é redundante, portanto pode ser retirado (Aalst, 1995).

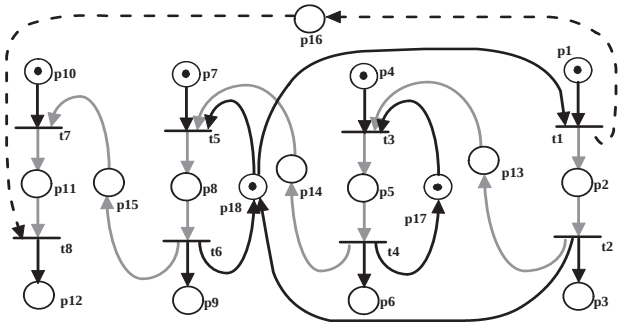



**Figura 3.8**: Problema de Escalonamento com Precedências Redundantes

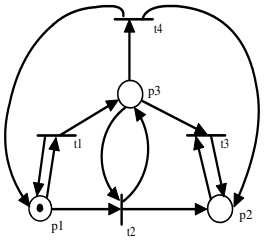



**Figura 3.9**: Rede de Petri não viva

Neste trabalho, a análise de invariância será usada para encontrar precedências redundantes e/ou conflitantes e ainda, para verificar a consistência e a conservação da rede, através de um algoritmo para a verificação das propriedades estruturais de conservação e de consistência em uma rede de Petri e que será discutido no capítulo 5.

## 3.3 Armadilhas e Sifões

Uma *armadilha* é um conjunto de lugares tal que qualquer transição de entrada dos lugares do conjunto, também é saída de algum desses lugares. Isto significa que se algum dos lugares da *armadilha* tem uma ficha então sempre existirá uma ficha em algum dos lugares da *armadilha*. O disparo das transições podem movimentar as fichas nos lugares da *armadilha* mas não pode retirar fichas dela.

Um *sifão*, também chamado de impasse estrutural (*structural deadlock*), ou simplesmente impasse, é um conjunto de lugares tal que qualquer transição de saída de um desses lugares também é entrada de algum desses lugares. Uma vez que todos os lugares no sifão tornem-se desmarcados, não haverá mais nenhuma ficha em qualquer lugar do sifão. Logo, um sifão que perca todas as suas fichas não poderá disparar nunca mais (DESROCHERS & AL-JAAR, 1995).

Na figura 3.7, os lugares $p_2$, $p_{13}$, $p_5$, $p_{14}$, $p_8$, $p_{15}$, $p_1$ e $p_{16}$ formam uma armadilha, por exemplo. Neste caso, como nenhum lugar tem fichas, este conjunto de fichas permanecerá sem fichas.

A fig. 3.9 mostra uma rede de Petri não viva. Os $p_1$, $p_2$ e $p_3$ formam um sifão, com o disparo de $t_1$, $t_2$ e $t_3$ provocando um impasse.

Segundo Desrochers & Al-Jaar (1995), armadilhas e sifões pode ser relacionados à p-invariância da seguinte forma:

**Propriedade 3.4** *Se $x$ é um p-invariante, o suporte de $x$ (definido na seção 3.2.3), $\|x\|$, é uma armadilha e um sifão.*

Na próxima seção, sifões e armadilhas serão usados no estabelecimento de propriedades de algumas subclasses de redes de Petri.

## 3.4 Subclasses de Redes de Petri Ordinárias

As subclasses de redes de Petri ordinárias foram definidas para diferenciar algumas restrições na estrutura das redes ordinárias. Por esta razão é muito fácil reconhecer a qual subclasse uma rede de Petri ordinária pertence e a partir daí estudar seu comportamento. Em muitos casos, propriedades como vivacidade e reversibilidade são completamente caracterizadas, graças a estas estruturas resultantes (Murata, 1989; Di Cesare *et. al.*, 1993). Antes de descrever estas subclasses são necessárias algumas definições de teoria dos grafos que serão mostradas a seguir(uma pequena introdução de teoria de grafos e além de mais algumas definições podem ser vistas no apêndice A).

Um *caminho elementar* é uma seqüência de nós, $x_1 x_2 \dots x_n,\ n \geqslant 1$, tal que:

1. $\forall i < n$, existe um arco direcionado de $i$ para $i+1$, $(x_i, x_{i+1})$;
2. não há repetição de nós na seqüência, isto é, $\forall i \neq j,\ x_i \neq x_j$

Quando $i > 1$, o caminho elementar é denotado por $EP(x_1, x_n)$. Se n=1 a seqüência é constituída por apenas um nó, sendo denotada por $x_1$.

Um *circuito elementar* é uma seqüência de nós, $x_1 x_2 \dots x_n,\ n > 1$, tal que:

1. $x_1, \dots x_{n-1}$ é um caminho elementar;
2. $x_1 = x_n$ e existe o arco $(x_n, x_1)$.

O circuito elementar é denotado por $EC(x_n)$.

Por conveniência, $EP(x_1, x_n)$ é interpretado como sendo um conjunto de nós, isto é, $\{x_1, x_2, \dots, x_n\}$ ($EC(x_n)$ também é tratado como sendo um conjunto de nós). Portanto, $x \in EP(x_1, x_n)$ significa que $x$ é um nó do caminho elementar entre $x_1$ e $x_n$. Dados $x, y \in P \cup T$, $EP(x, y)$ pode representar qualquer caminho elementar ou o conjunto de todos os caminhos elementares entre $x$ e $y$ em um contexto particular, e pode ser usado como um conjunto. Se não existe tal caminho entre $x$ e $y$, então pode-se dizer que $EP(x, y) = \oslash$.

Também é necessária para a apresentação das propriedades das subclasses de RdP, a definição de redes de Petri conectadas e fortemente conectadas:

**Definição 3.4** *Uma rede de Petri é conexa (ou conectada) se para qualquer par de nós $u$ e $v$ (lugares e/ou transições) existe um cadeia (definida no apêndice A) entre $u$ e $v$.*

**Definição 3.5** *Uma rede de Petri é fortemente conexa (ou fortemente conectada) se para qualquer par de nós $u$ e $v$ (lugares e/ou transições) existe um caminho de $u$ para $v$ e outro de $v$ para $u$.*

Segundo DiCesare *et. al.* (1993) se uma rede de Petri $N$ é conexa, consistente e conservativa então ela é fortemente conexa. Donde, se uma RdP não for fortemente conexa, ela não será conexa e/ou não será consistente e/ou não será conservativa.

- **Grafo de Eventos – GE**

  Em um grafo de eventos cada lugar tem exatamente uma transição de entrada e uma de saída.

  **Definição 3.6** *Um Grafo de Eventos é uma Rede de Petri* $N =< P, T, I, O >$ *tal que para cada* $p_i \in P, |I(p_i)| = \left|\left\{t_j \mid p_i \in O(t_j)\right\}\right| = 1 \, e \, |O(p_i)| = \left|\left\{t_j \mid p_i \in I(t_j)\right\}\right| = 1.$

  Por exemplo, a rede da figura 3.10a é um grafo de eventos.

  Em um grafo de eventos não existem conflitos, já que cada lugar tem exatamente uma transição de saída. Em geral, existem sincronizações que correspondem a transições com diversos lugares de entrada. Sendo assim os grafos de eventos são bem adaptados para modelar sistemas cujo comportamento qualitativo é determinístico.

  Um grafo de eventos fortemente conexo tem as seguintes propriedades (CARDOSO & VALETTE, 1997):

  1. cada circuito elementar de um grafo de eventos forma um e um único componente conservativo (um componente conservativo é o conjunto de lugares correspondentes a elementos não nulos em um p-invariante, conforme seção 3.2.3);
  2. um grafo de eventos forma um único componente repetitivo estacionário (um componente repetitivo estacionário é um conjunto de transições correspondentes a elementos não nulos em um t-invariante, conforme seção 3.2.3);
  3. um grafo de eventos marcado fortemente conexo é vivo se e somente se existe no mínimo uma ficha em cada circuito elementar;
  4. um grafo de eventos marcado vivo é seguro se e somente se todo lugar no grafo de eventos pertence a um circuito com exatamente uma ficha.

- **Máquinas de Estado – ME**

  Em uma máquina de estados cada transição tem exatamente um lugar de entrada e um lugar de saída.

  **Definição 3.7** *Uma Máquina de Estado é uma Rede de Petri* $N =< P, T, I, O >$ *tal que para todo* $t_j \in T, \left|I(t_j)\right| = 1 \, e \, \left|O(t_j)\right| = 1.$

  Em uma máquina de estado não há sincronização e em geral há conflitos. Uma máquina de estado fortemente conexa (figura 3.10b) possui as seguintes propriedades (CARDOSO & VALETTE, 1997):

  1. cada circuito elementar de uma máquina de estado forma um componente repetitivo estacionário;
  2. uma máquina de estado forma um e um único componente conservativo;

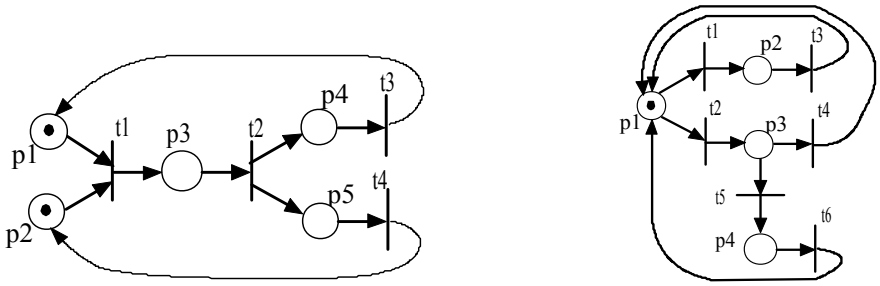



**Figura 3.10**: (a) Grafo de Eventos (b) Máquina de Estado

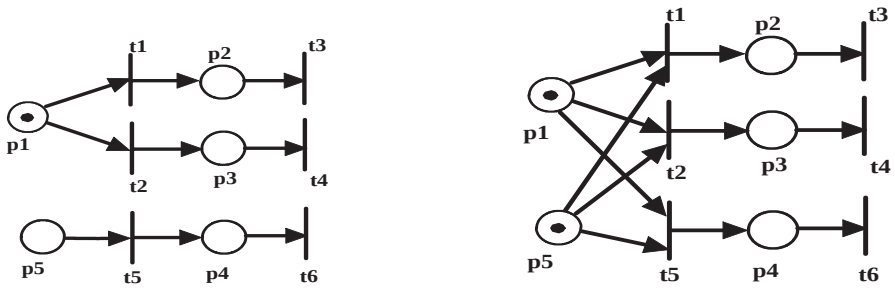



**Figura 3.11**: (a) Redes de Escolha Livre (b) Redes de Escolha Livre Extendida

3. uma máquina de estado é viva e reiniciável se e somente se existe no mínimo uma ficha na marcação inicial da rede.

- **Rede de Escolha Livre – REL**

  Em uma Rede de Escolha Livre, se um lugar for a entrada de mais de uma transição (conflito potencial), então ele é o único lugar de entrada destas transições conforme a definição 3.7.

  A rede mostrada pela figura 3.11a é exemplo de uma rede de escolha livre.

  **Definição 3.8** *Uma Rede de Escolha Livre é uma Rede de Petri* $N =< P, T, I, O >$ *tal que para todo* $t_j \in T$ *e para todo* $p_i \in I(t_j)$*, ou* $\left|I(t_j)\right| = \{p_i\}$ *e* $|O(p_i)| = \left\{t_j\right\}$.

  A importância desta definição é a forma pela qual ela permite controlar um conflito (PETERSON, 1981), pois ou todas as transições em conflito estão habilitadas simultâneamente, ou nenhuma delas está. Isto permite que a escolha seja feita livremente independente da marcação dos outros lugares.

  Esta subclasse permite modelar as relações de sequenciamento, de escolha e de concorrência, com a ressalva de que sincronizações e escolhas não ocorram na mesma transição.

  Hack (1972) estabeleceu as condições necessárias e suficientes para que redes de escolha livre marcadas sejam vivas e seguras.

Hack provou ainda que a condição necessária e suficiente para vivacidade em uma rede de escolha livre marcada é a de que todo *sifão* deve conter uma *armadilha* marcada. As condições necessárias e suficientes para segurança envolvem a demonstração de que a rede de Petri de escolha livre está coberta por uma união de máquinas de estado (HACK, 1972).

Segundo DiCesare *et. al.*(1993), é possível relacionar-se as propriedades estruturais às redes de escolha livre chegando ao resultado mostrado a seguir[3]:

**Propriedade 3.5** *Seja C a matriz de incidência de uma rede de escolha livre* $N =< P, T, I, O >$ *conexa.* $N$ *é estruturalmente viva e estruturalmente limitada se e somente se:*

$$\begin{aligned} &\exists y > 0, \quad \textit{tal que} \quad C \cdot y = 0, \quad \textit{isto é,}\ N\ \textit{é consistente e,} \\ &\exists x > 0, \quad \textit{tal que} \quad x^t \cdot C = 0, \quad \textit{isto é,}\ N\ \textit{é conservativa e,} \\ &posto(C) \quad = \quad |T| + |P| - a - 1, \end{aligned} \tag{3.18}$$

*onde* $a$ *é o número de arcos na relação de fluxo dada pela matriz* $I$.

Com relação à propriedade 3.1 a hipótese de que a rede é limitada e viva agora é uma condição necessária e suficiente para a ocorrência da consistência e da conservação na rede. Além disso, a condição do posto é uma igualdade, e $\delta$ é substituído por este valor em redes de Petri de escolha livre: $\delta = a - |P|$.

Note que máquinas de estado e grafos de eventos são casos particulares de redes de escolha livre, logo a propriedade 3.5 pode ser estendida a estes dois casos.

Outro resultado importante dado por DiCesare *et. al.*(1993) caracteriza as propriedades de limitação, vivacidade e reversibilidade para redes de escolha livre:

**Propriedade 3.6** *Seja a rede de Petri* $N =< P, T, I, O, m_0 >$*, uma rede de escolha livre viva e limitada. Esta rede é reversível se todas as armadilhas estiverem marcadas em sua marcação inicial* $m_0$.

- **Rede de Escolha Livre Extendida – RELE**

  Em uma Rede de Escolha Livre Extendida, se dois lugares tem alguma transição de saída em comum, então eles tem todas as transições de saída em comum, como mostrado na figura 3.11b.

- **Rede de Petri Simples – RPS**

  Hack (1972) definiu uma outra subclasse de redes de Petri denominada Rede de Petri Simples. Neste tipo de rede de Petri cada transição tem no máximo um lugar de entrada compartilhado com outra transição, como mostrado na figura 3.12. Esta exigência serve para restringir a forma em que os conflitos ocorrem. Isto porque em algumas situações podem ocorrer uma mistura de concorrência e conflito, o que leva à confusão, mostrada na seção 2.3.2 e

[3] DiCesare *et. al.*(1993), pág. 54

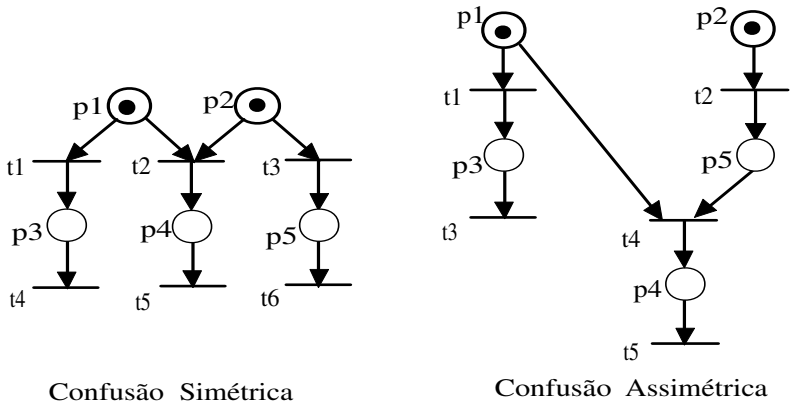



**Figura 3.12**: Exemplos de confusão

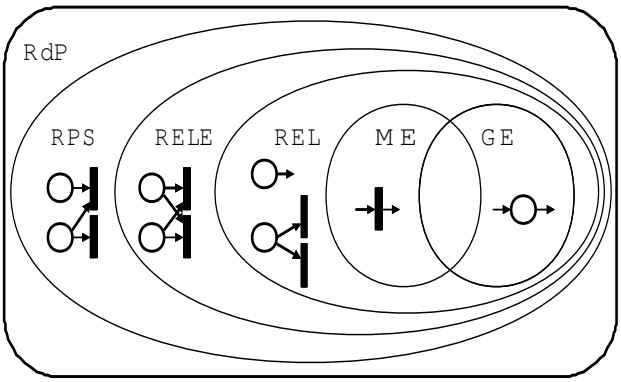



**Figura 3.13**: Subclasses de Redes de Petri

também exemplificada pela figura 3.12. Tanto a confusão simétrica quanto a assimétrica não satisfazem a definição de grafo de eventos, ou de máquina de estados, ou ainda, de rede de escolha livre (DESROCHERS & AL-JAAR, 1995).

Na confusão simétrica mostrada pela figura 3.12, as transições $t_1$ e $t_3$ são concorrentes (ambas estão habilitadas e podem disparar) e ao mesmo tempo estas transições estão em conflito com $t_2$. Na confusão assimétrica mostrada pela figura 3.12, $t_1$ e $t_2$ são concorrentes, $t_1$ e $t_4$ não estão em conflito, mas se ocorrer o disparo de $t_2$, $t_1$ e $t_4$ ficarão em conflito.

**Definição 3.9** *Uma Rede de Petri Simples é uma Rede de Petri* $N =< P, T, I, O >$ *tal que cada transição tem no máximo um lugar de entrada compartilhado com outra transição.*

Esta subclasse permite a modelagem das relações de sequenciamento, de escolha e de concorrência, porém geralmente as escolhas não são livres mas podem ser resolvidas localmente (DICESARE *et. al.* , 1993).

Um rede de Petri simples também é conhecida como uma rede de escolha assimétrica.

Uma rede de Petri Simples marcada é viva se todo impasse (sifão) contêm uma armadilha marcada (condição necessária mas não suficiente).

Um exemplo típico de redes de Petri simples é o modelo de um sistema em que um recurso é compartilhado por dois ou mais usuários, como mostrado pela figura 2.10.

A figura 3.13 mostra o relacionamento entre estas subclasses. Esta classificação de redes de Petri será utilizada na próxima seção, para identificar os processos definidos na seção 2.4[4] com estas subclasses.

### 3.4.1 Resumo das Propriedades de Redes de Petri

Nesta seção serão apresentadas novamente as propriedades mostradas na seção anterior, mas de forma resumida.

- Um lugar $p \in P$ é k-limitado se $\forall m \in R(N, m_0),\ \ \exists k \in \mathbb{N} \mid m(p) \leqslant k$.
- $N$ é k-limitada se $p$ é k-limitado, $\forall p \in N$.
- $N$ é segura se se ela é 1-limitada.
- $N$ é viva se a partir de qualquer estado alcançável $m \in R(N, m_0)$ e $\forall t \in T$, existe uma seqüência de disparos $\sigma$ que habilita $t$.
- Uma marcação $m'$ é um estado–base (*homestate*) se para cada marcação $m \in R(N, m_0)$, $m'$ é alcançável a partir de $m$.
- $N$ é reversível se $m_0$ é um estado–base.
- Vivacidade Estrutural – Uma rede de Petri $N$ é dita ser *estruturalmente viva* se existe uma marcação inicial para a qual $N$ é viva.
- Controlabilidade – Uma rede de Petri $N$ é dita ser *completamente controlável* se uma marcação qualquer é alcançada a partir de qualquer outra marcação.
- Limitação Estrutural – Uma rede de Petri $N$ é denominada *estruturalmente limitada* se ela é limitada para qualquer marcação inicial finita $m_0$.
- Conservação – Uma rede de Petri $N$ viva é conservativa (parcialmente) se e somente se existe um vetor $x$ de inteiros positivos (não-negativos) tal que $x^t \cdot C = 0,\ x \neq 0$.
- Repetitividade – Segundo Murata (1989), uma rede de Petri $N$ com $s$ lugares é (parcialmente) repetitiva se e somente se existe um vetor $y$ de inteiros positivos (não-negativos) tal que $C \cdot y \geqslant 0, \quad y \neq 0$.
- Consistência – Segundo Murata (1989), uma rede de Petri $N$ com $s$ lugares é (parcialmente) consistente se e somente se existe um vetor $y$ de inteiros positivos (não-negativos) tal que $C \cdot y = 0, \quad y \neq 0$. Caso contrário, a rede de Petri é inconsistente.
- Um sistema é consistente (inconsistente) se sua modelagem em Rede de Petri é consistente (inconsistente).

[4]processos simples, com compartilhamento, com alternância e com sincronismo

- Um P-invariante (lugar invariante) é um vetor de inteiros positivos $x$ de dimensão $s$ que satisfaz a equação $x^t.C = 0$.
- O conjunto de lugares correspondentes a elementos não nulos em um p-invariante $x \geqslant 0$ é denominado *suporte deste invariante* ou *componente conservativo* e é denotado por $\|x\|$. Um suporte é *minimal* se não existe outro invariante $x_1$ tal que $x_1(t) \leqslant x(t)$ para todo $t$.
- Um T-invariante é um vetor de inteiros positivos $y$ de dimensão $r$ que satisfaz a equação $C.y = 0$. Seja $\sigma$ uma seqüência de disparo e $S$ seu vetor característico. Sendo $S = y$ um T-invariante, e sendo $m_0$ uma marcação inicial de $N$ e $m$ uma marcação alcançável a partir de $m_0$ tem-se $m = m_0$.
- O conjunto de transições correspondentes a elementos não nulos em um t-invariante $y \geqslant 0$ é denominado *suporte deste invariante* ou *componente repetitivo estacionário* e é denotado por $\|y\|$. Um suporte é *minimal* se não existe outro invariante $y_1$ tal que $y_1(p) \leqslant y(p)$ para todo $p$.
- Segundo DiCesare *et. al.* (1993) se uma rede de Petri $N$ é conexa, consistente e conservativa então ela é fortemente conexa.
- Um grafo de eventos fortemente conexo tem as seguintes propriedades (Cardoso & Valette, 1997):
    1. cada circuito elementar de um grafo de eventos forma um e um único componente conservativo;
    2. um grafo de eventos forma um único componente repetitivo estacionário;
    3. um grafo de eventos marcado fortemente conexo é vivo se e somente se existe no mínimo uma ficha em cada circuito elementar;
    4. um grafo de eventos marcado vivo é seguro se e somente se todo lugar no grafo de eventos pertence a um circuito com exatamente uma ficha.
- Uma máquina de estado fortemente conexa (figura 3.10b) possui as seguintes propriedades (Cardoso & Valette, 1997):
    1. cada circuito elementar de uma máquina de estado forma um componente repetitivo estacionário;
    2. uma máquina de estado forma um e um único componente conservativo;
    3. uma máquina de estado é viva e reiniciável se e somente se existe no mínimo uma ficha na marcação inicial da rede.
- Uma rede de Petri Simples marcada é viva se todo impasse (sifão) contêm uma armadilha marcada (condição necessária mas não suficiente).
- **Propriedade 3.3:** Dada a rede de Petri $N =< P, T, I, O >$ e sua matriz de incidência $C$, se a equação $Cy = 0$ só admite a solução trivial, então $N$ não é reversível. Esta é uma condição suficiente, mas não é necessária para a irreversibilidade. Donde, a solução não trivial garante apenas a reversibilidade parcial.

- **Propriedade 3.4:** Se x é um p-invariante, o suporte de $\|x\|$ é uma armadilha e um sifão.
- **Propriedade 3.6** Seja a rede de Petri $N =< P, T, I, O, m_0 >$, uma rede de escolha livre viva e limitada. Esta rede é reversível se todas as armadilhas estiverem marcadas em sua marcação inicial $m_0$.

## 3.5 Associação dos Processos às Subclasses

Os processos definidos na seção 2.4 podem ser associados às subclasses de redes de Petri ordinárias da seguinte forma:

- Processos Ordinários – se forem fortemente conexos, são tanto grafos de eventos, pois não possuem conflitos em sua estrutura; quanto são também máquinas de estado, pois não possuem sincronização em sua estrutura.
- Processos com Sincronismo – se forem fortemente conexos, são grafos de eventos, pois não possuem conflitos em sua estrutura e incluem sincronismo.
- Processos com Alternância – se forem fortemente conexos, são máquinas de estado, pois não possuem sincronização em sua estrutura, mas incluem conflito.
- Processos com Compartilhamento – se forem fortemente conexos, são redes de Petri simples, quando forem compostos de processos simples ou com sincronismo, pois os únicos lugares de conflito são os de exclusão mútua. Quando forem compostos de processos com alternância também serão redes de Petri simples se cada transição da rede tiver no máximo um lugar de entrada compartilhado com outra transição.

Como foi visto na seção 3.4, se uma rede de Petri não for fortemente conexa, ela não será conexa ou não será consistente ou não será conservativa. Portanto, os processos que serão estudados neste trabalho deverão ser fortemente conexos.

### 3.5.1 Análise Estrutural de Processos

A seguir algumas propriedades dos processos serão apresentadas:

**Processos com Sincronismo** – Pela seção 3.5 sabe-se que os processos com sincronismo são casos particulares de grafos de eventos. Logo, os processos com sincronismo terão as mesmas propriedades dos grafos de eventos. Desta forma, um processo com sincronismo fortemente conexo tem as seguintes propriedades:

1. cada um dos seus circuitos elementares forma um e um único componente conservativo;
2. o processo com sincronismo forma um e um único componente repetitivo estacionário;
3. o processo com sincronismo é vivo se e somente se existe no mínimo uma ficha em cada circuito elementar;

4. o processo com sincronismo vivo é seguro se todo lugar do processo com sincronismo pertence a algum circuito elementar com exatamente uma ficha.

**Processos com Alternância** – Pela seção 3.5 sabe-se que os processos com alternância são casos particulares de máquinas de estado. Logo, os processos com alternância terão as mesmas propriedades das máquinas de estado. Desta forma, um processo com alternância fortemente conexo tem as seguintes propriedades:

1. cada um dos seus circuitos elementares forma um componente repetitivo estacionário;
2. o processo com alternância forma um e um único componente conservativo;
3. o processo com alternância é vivo e reiniciável se existe no mínimo uma ficha em seu componente conservativo e se este componente for fortemente conexo.

**Processos Ordinários** – Note pela definição de processos ordinários, de processos com sincronismo e de processos com alternância (seção 2.4), processos ordinários são casos particulares tanto de processos com sincronismo quanto de processos com alternância. E ainda, um processo ordinário é composto de um único circuito elementar. Desta forma, um processo ordinário fortemente conexo possue as seguintes propriedades:

1. o processo ordinário forma um e um único componente conservativo;
2. o processo ordinário forma um e um único componente repetitivo estacionário;
3. o processo ordinário é vivo se e somente se existe no mínimo uma ficha em seu circuito elementar.

**Processos com Compartilhamento** – Pela seção 2.4, sabe-se que Processos com Compartilhamento são processos que compartilham algum recurso. Eliminando-se os lugares de exclusão mútua e seus respectivos arcos, os processos restantes serão uma combinação de processos ordinários, com sincronismo e/ou com alternância. A combinação destes processos será vista no capítulo 5. O caso mais simples de processos com compartilhamento é representado por uma estrutura de exclusão mútua, composta de um lugar que representa um recurso compartilhado e de operações (representadas por lugares) que compartilham este recurso, conforme a definição 2.21.

Contudo, existem processos com compartilhamento mais complexos que podem não apresentar as propriedades acima. A análise a seguir considera este tipo de processo.

A estrutura representada desta forma é uma máquina de estados fortemente conexa e tem as seguintes propriedades:

1. cada circuito elementar do processo forma um componente repetitivo estacionário;
2. o processo com compartilhamento forma um e um único componente conservativo;
3. o processo com compartilhamento é vivo e reiniciável se existe no mínimo uma ficha em seu componente conservativo.

É um fato conhecido que um lugar, em uma RdP, pode representar uma sub-rede, assim como uma sub-rede pode ser representada por um único lugar.

Donde, se em um processo com compartilhamento simples, um lugar que representa uma operação, for substituído por uma sub-rede, este processo continuará a ser consistente e conservativo, se o circuito, formado pela sub-rede e o lugar que representa o recurso compartilhado, continuar sendo consistente e conservativo.

A figura 3.14 mostra uma substituição possível no processo com compartilhamento que mantêm a consistência e a conservação da rede. Já a figura 3.15 mostra uma substituição que torna a rede inconsistência e não conservativa.

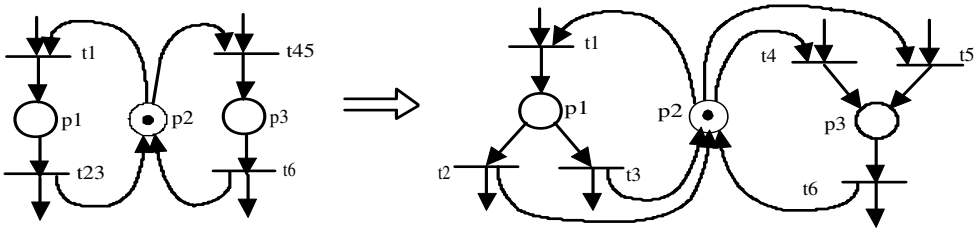



**Figura 3.14**: Processo com Compartilhamento Consistente e Conservativo

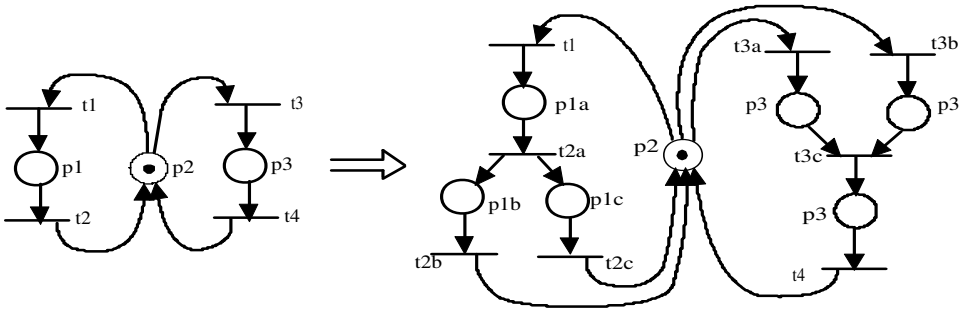



**Figura 3.15**: Processo com Compartilhamento Inconsistente e Não Conservativo

Portanto, cada processo com compartilhamento deve ser analisado sob o ponto de vista de consistência e conservatividade.

Outros casos envolvendo processos com compartilhamento serão discutidos no capítulo 5.

### 3.5.2 Sumário da Tipificação dos Processos

A caracterização dos componentes conservativos e repetitivos estacionários dos processos permite fazer um método para a transposição da modelagem de redes de Petri para a modelagem em programação linear inteira mista. Este método será mostrado no cap. 5.

A tabela 3.1 resume os conceitos mostrados nesta seção.

## 3.6 Redes de Petri e Sistemas a Eventos Discretos Periódicos

Conforme estabelecido na seção 2.5 este trabalho visa modelar e controlar um sistema a eventos discretos periódicos a partir de redes de Petri e de programação linear inteira mista.

| Processos | Componente Conservativo | Componente Repetitivo | Propriedades |
|---|---|---|---|
| Processo Ordinário | forma um e um único componente conservativo | forma um e um único componente repetitivo | é vivo se e somente se existe no mínimo uma ficha neste processo |
| Processo com Sincronismo | cada circuito elemetar forma um componente conservativo | forma um e um único componente repetitivo | é vivo se e somente se existe no mínimo uma ficha em cada circuito elementar |
| Processo com Alternância | forma um e um único componente conservativo | cada circuito elementar forma um componente repetitivo estacionário | é vivo e reiniciável se existe no mínimo uma ficha em seu componente conservativo e se ele for fortemente conexo |
| Processo com Compartilhamento Simples | forma um e um único componente conservativo | cada circuito elementar forma um componente repetitivo estacionário | é vivo e reiniciável se existe no mínimo uma ficha em seu componente conservativo e se ele for fortemente conexo |

**Tabela 3.1**: Tipificação dos Processos

Dentre as possíveis classificações de rede de Petri apresentadas na seção 3.1, são relevantes para este objetivo as redes de Petri estendidas temporizadas.

O diagrama da figura 3.16 esquematiza, dentro deste conjunto, as diversas possibilidades quanto à consistência, conservatividade e tipo de temporização. A definição dada a seguir tem como conseqüência o fato de que as redes de Petri que podem ser tratadas neste trabalho correspondem às áreas hachuradas da figura 3.16.

De fato é imediato verificar a necessidade das condições de conservatividade e consistência, assim como de outras condições, pois as redes de Petri que modelam sistemas a eventos discretos periódicos devem:

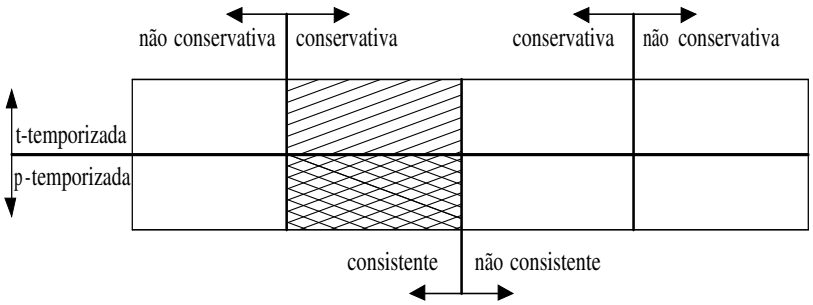



**Figura 3.16**: Redes de Petri que podem ser Tratadas

1. ser consistentes e reversíveis, caso contrário os sistemas a serem tratados não poderão ter um funcionamento cíclico;
2. ser estruturalmente vivas, caso contrário os sistemas a serem tratados não serão inicializados, ou se puderem ser iniciados, existirão bloqueios ou impasses;
3. ser conservativas, caso contrário a rede não é limitada, e portanto não é reversível;
4. conter um e um único lugar marcado em cada um de seus componentes conservativos, sendo que estes lugares não marcados devem ser não temporizados, caso contrário, não será possível modelá-lo em programação linear inteira mista;
5. ser temporizadas com tempo determinístico, caso contrário não será possível o controle do sistema através da modelagem que será proposta no cap. 4, pois todos os lugares e transições terão temporização nula ou terão temporizações não conhecidas *a priori*, o que impossibilitaria determinar os intantes de disparos das transições para o escalonamento feito através da modelagem.

Se um componente conservativo contiver mais de um lugar marcado ou se contiver um lugar marcado temporizado, deve ser possível através de alguns disparos de transições da rede mudar a marcação inicial destes lugares colocando as fichas apenas em lugares não temporizados, de tal forma que ao final a rede contenha apenas lugares–não–marcados–temporizados e lugares–marcados–não– temporizados, e com um e um único lugar marcado em cada um de seus componentes conservativos.

Se não for possível fazer esta mudança em um componente conservativo porque o mesmo não possui lugares não temporizados, então deve-se transformar o lugar marcado temporizado em uma sequência (lugar não marcado temporizado) – (lugar marcado não temporizado) (esta regra de transformação será apresentada no capítulo 5), mas se não for possível fazer a mudança por qualquer outra hipótese a rede não poderá ser modelada pela modelagem em programação linear inteira mista.

Na figura 3.16, as redes de Petri temporizadas são divididas em 2 grupos, as p-temporizadas e as t-temporizadas e, como sempre é possível transformar uma RdP t-temporizada em um p-temporizada, as p-temporizadas tem um destaque maior, pois é a partir de redes de Petri p-temporizadas que a modelagem em programação linear inteira mista será realizada, como será visto no próximo capítulo.

Com isto a seguinte definição é proposta :

**Definição 3.10** *Uma Rede de Petri é denominada Rede de Petri de Sistemas a Eventos Discretos Periódicos, se for temporizada, determinística, reversível, conservativa, consistente, estruturalmente viva, se contiver apenas um único lugar marcado em cada um de seus componentes conservativos e se seus lugares marcados forem não temporizados.*

A verificação das propriedades de consistência e de conservação pode ser realizada através da análise de invariância. Esta análise possui algoritmos que não geram explosão combinatorial, como é o caso do algoritmo que será estudado no cap. 5. Este algoritmo será usado para encontrar os componentes conservativos e repetitivos das redes de Petri de sistemas a eventos discretos e através destes componentes será possível também verificar se cada um destes componentes contém um e um único lugar marcado. Estes componentes serão utilizados na modelagem proposta no próximo capítulo. Porém a verificação da reversabilidade só ser feita em alguns casos através da análise enumerativa, como será visto no capítulo 5.

A verificação da temporização dos elementos da rede será feita por regras apresentadas no capítulo 5.

No cap. 4 será proposta uma modelagem em programação linear inteira mista para sistemas a eventos discretos periódicos descritos pelas redes de Petri de sistemas a eventos discretos.

As redes de Petri que não forem conservativas ou consistentes, ou que forem estocásticas ou autônomas não poderão ser tratadas pelo modelo em MILP. Estas redes serão consideradas não tratáveis pela modelagem em MILP.

## 3.7 Observações Finais

Neste capítulo, foram estudadas as classes, subclasses, principais propriedades e características de redes de Petri que são usadas para modelar sistemas a eventos discretos, a fim de especificar quais destes modelos poderiam ser tratados pela modelagem em MILP (cap. 4).

As subclasses de redes de Petri ordinárias estudadas, foram associadas aos processos, o que possibilitou a análise estrutural de alguns destes processos.

Foi visto na seção 3.6 que apenas as redes de Petri temporizadas com tempo determinístico poderão ser tratadas pela modelagem que será proposta no próximo capítulo. Deve-se ressaltar que a topologia das RdP temporizadas é igual a topologia de RdP ordinárias, por isso as RdP temporizadas que serão estudadas poderão ser grafos de eventos, máquinas de estado, RdP de escolha livre, RdP de escolha livre extendida e RdP Simples.

O próximo capítulo irá tratar da modelagem matemática em MILP para as redes de Petri de sistemas a eventos discretos periódicos.

# 4

# Modelagem Matemática de Sistemas a Eventos Discretos Periódicos

O objetivo principal deste capítulo é a proposta de uma modelagem em programação linear inteira mista (MILP) para sistemas a eventos discretos periódicos cujo comportamento possa ser descrito por redes de Petri de sistemas a eventos discretos periódicos.

Antes de apresentar a modelagem é necessário identificar os elementos em torno dos quais a formulação MILP será feita. Esta identificação baseia-se na proposta de Zhou & DiCesare (1993) e leva a uma distinção entre os lugares da rede de Petri associada.

A terminologia de sistemas de manufatura será adotada aqui para garantir uma melhor compreensão dos conceitos envolvidos na passagem da modelagem em rede Petri para a modelagem em MILP. Esta adoção não limita a classe de redes de Petri que pode ser modelada pela formulação em MILP, apenas os elementos da rede de Petri serão interpretados sob o ponto de vista de sistemas de manufatura.

## 4.1 Funcionamento Periódico de um SED Periódico

De modo geral nos sistemas a eventos discretos existem 3 tipos de eventos: os de entrada, os de saída e os internos. Ao modelar via redes de Petri esta situação é mapeada (representada) por transições de entrada e de saída, definidas na seção 2.3.1, e por transições internas, respectivamente, da seguinte forma:

- transição de origem ou de entrada – possui lugares de saída e nenhum lugar de entrada;
- transição de destino ou de saída – possui lugares de entrada e nenhum lugar de saída
- transição interna – possui pelo menos um lugar de entrada e um de saída;

Note que uma transição de origem está incondicionalmente habilitada, e que uma transição de destino consome fichas, mas não as produz.

Usualmente as transições de entrada representam a chegada de matéria-prima ou o início de um processo, as transições de saída representam a finalização de um produto ou de um processo e as transições internas representam o início ou o fim de execução de operações de processos.

Quando se deseja um funcionamento periódico para um SED, é mister condicionar os disparos das transições de entrada a partir dos disparos das transições de saída. Formalmente o funcionamento periódico do sistema pode ser verificado pela análise de consistência, pois se o sistema não for consistente, então o sistema não será repetitivo estacionário e de acordo com a propriedade 3.3 (ver seção 3.2.3) também não será reversível. Este é o caso das redes de Petri mostradas pela figura 4.1. Esta figura mostra 2 topologias possíveis para redes de Petri não repetitivas.

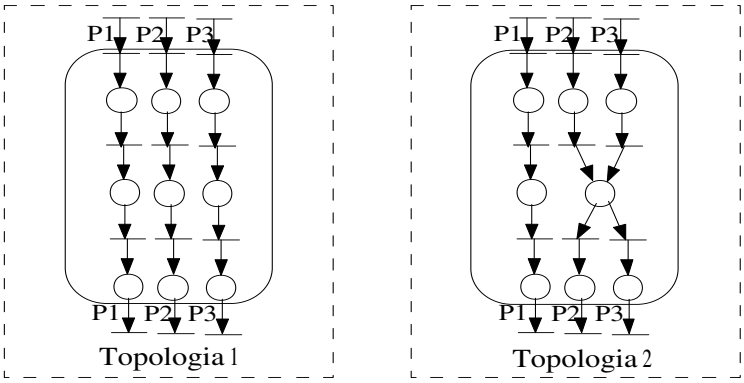



**Figura 4.1**: Topologias Possíveis para RdP não repetitivas

Para resolver este tipo de problema, neste trabalho, vamos assumir que as transições de saída podem ser interconectadas às transições de entrada através de lugares marcados (que serão classificados na seção 4.2.1 como C-lugares).

A figura 4.2 mostra um exemplo da inclusão destes lugares a redes de Petri não repetitivas, e como existem inúmeras possibilidades para a inclusão destes lugares, esta figura mostra duas possibilidades diferentes de configuração para que estes lugares modelem a repetibilidade de um sistema composto de três processos.

Considere apenas a inclusão dos C-lugares nas redes de Petri da figura 4.2:

- na 1ª configuração, os processos P1, P2 e P3 podem ser realizados periodicamente, independentemente uns dos outros;
- porém na 2ª configuração, o processo P1 pode ser realizado periodicamente sendo independente dos outros processos P2 e P3, já estes dois processos competem pela utilização das fichas mas podem ser realizados periodicamente se forem executados com alternância.

A consistência, a conservação, a vivacidade e a reversibilidade da rede dependem ainda da topologia da rede de Petri na qual os C-lugares são incluídos. Esta dependência pode ser exemplificada pelas figuras 4.3 e 4.4 que mostram as 2 topologias da figura 4.1 associadas às configurações da figura 4.2.

Na figura 4.3, as configurações 1 e 2 são associadas à topologia 1, nestes dois casos a rede de Petri resultante é consistente, conservativa, viva e reversível. Deve-se ressaltar que para fazer a associação da topologia 1 à configuração 1, foi necessário adaptar a topologia à configuração, pois esta configuração possui 3 transições de entrada e a topologia 1 leva em consideração que a rede possua apenas 2 transições de entrada.

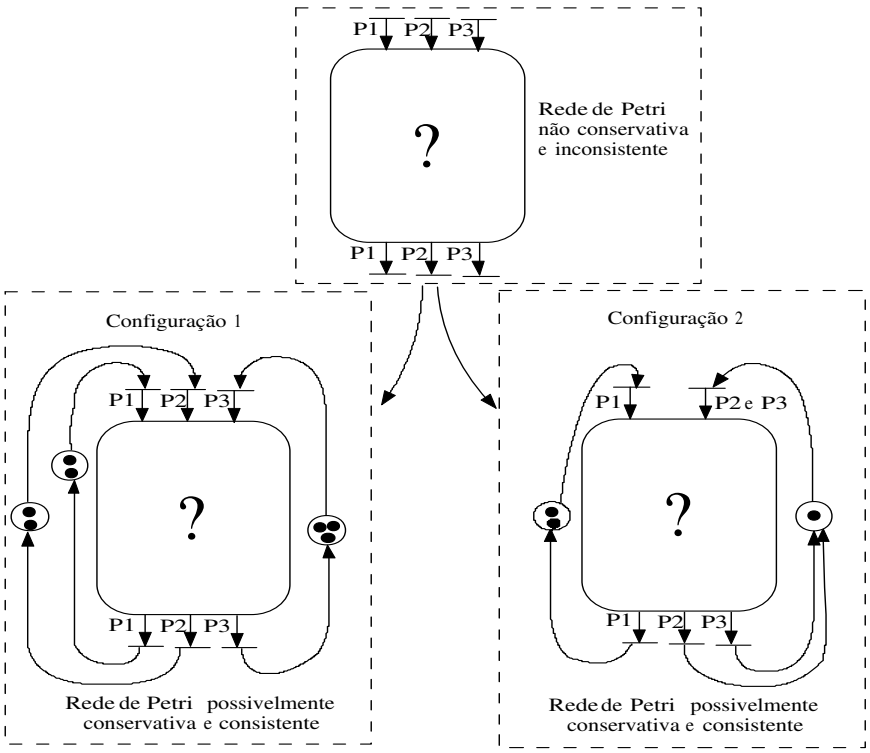



**Figura 4.2**: Redes de Petri Possivelmente Conservativas, Vivas e Reversíveis

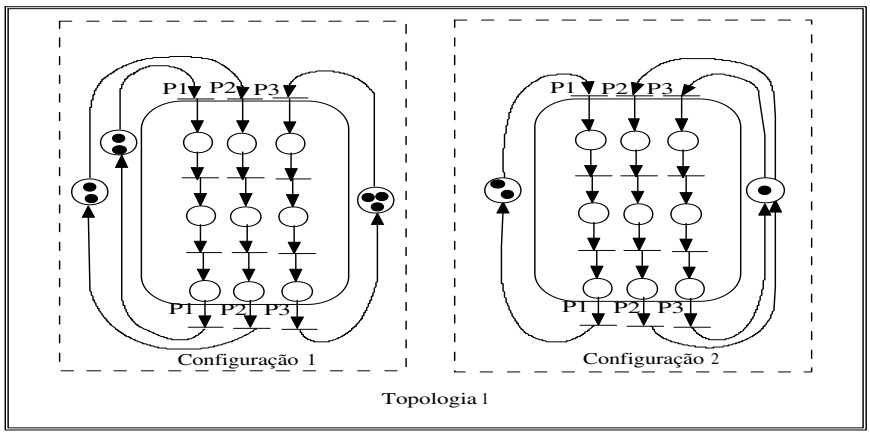



**Figura 4.3**: Redes de Petri Conservativas, Consistentes, Vivas e Reversíveis

Na figura 4.4, as configurações 1 e 2 são associadas à topologia 2, o que leva a RdP da configuração 1 a ser consistente, conservativa, porém, não–viva e não–reversível; e leva a RdP da configuração 2 a ser ilimitada, portanto inconsistente, não-conservativa e não-reversível. Neste caso é fácil verificar por inspeção, que a rede é viva.

É importante destacar que a maneira de interligar saídas e entradas depende da aplicação e deve ser definida pelo modelador. Contudo as ferramentas definidas no capítulo 3 permitem determinar se uma dada maneira de interligar leva ou não a um comportamento periódico com uma marcação inicial viva.

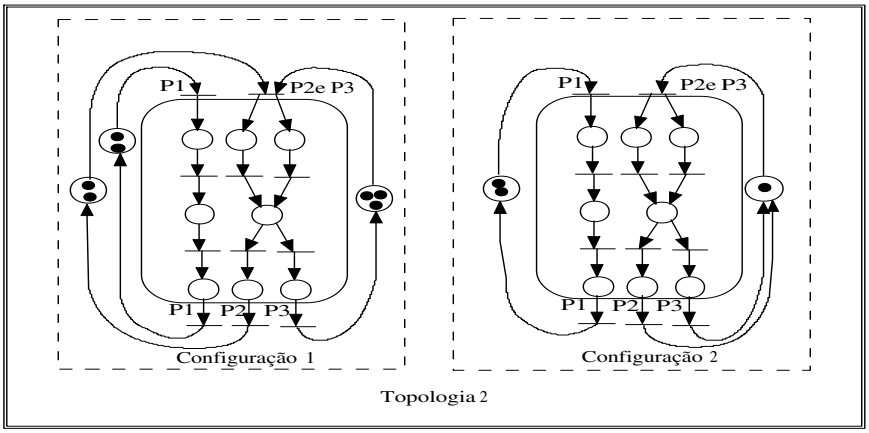



**Figura 4.4**: Redes de Petri Ilimitadas

## 4.2 Modelagem de Sistemas de Manufatura através de RdP

Um sistema de manufatura é um conjunto de tarefas (operações) que interagem com um conjunto de recursos e que resulta em um produto (ZHOU & DICESARE, 1993). As tarefas são os processos de manufatura incluindo atividades nas máquinas, transporte de materiais e processamento de informação, que devem ocorrer para se produzir algo. As pessoas (trabalhadores), as máquinas e as matérias-primas, que são requeridos para realizar estas tarefas são os recursos. Supõe-se conhecidas as relações de precedência entre as tarefas que constituem um processo e também os recursos necessários para sua realização.

A seguinte interpretação de lugares, transições e movimentação das fichas em Redes de Petri de sistemas de manufatura é dada por Zhou & DiCesare (1993):

- Um lugar representa um recurso ou uma operação.
- Se um lugar representa um recurso, uma ou mais fichas no lugar indicam que o recurso está disponível e a ausência de fichas indica o contrário, sendo portanto uma condição e como tal não pode existir nenhum tempo de duração associada a ela. Se um lugar representa uma operação, pode ser associada ao lugar o tempo de duração da operação, e ainda, uma ficha neste lugar indica que a operação está sendo executada e nenhuma ficha indica que ela não esta sendo realizada.
- Uma transição representa a ocorrência de um evento, como por exemplo, o início ou o fim de atividade de uma operação.

Esta interpretação será estendida para as redes de Petri de sistemas a eventos discretos periódicos neste trabalho.

### 4.2.1 Classificação de Lugares

Dada uma rede de Petri de sistemas a eventos discretos periódicos, é sempre possível classificar todos os seus lugares como sendo de um dentre três tipos de lugares:

**A-lugares** – representam tarefas (operações) com tempo de duração finito e fixo (conforme Zhou & DiCesare, 1993).

**B-lugar** – representa a disponibilidade de um recurso (p.ex. uma máquina). O tempo de duração associado a um B-lugar deve ser sempre nulo e sua marcação inicial deve ser igual ou maior que 1. B-lugares são componentes essenciais dos processos com compartilhamento (definição 2.25).

**C-lugar** – representa a quantidade de instâncias de um mesmo processo simples que podem ser realizadas ao mesmo tempo pelo sistema. Este lugar une a última tarefa com a primeira tarefa de um mesmo processo ordinário ou de um mesmo processo simples, criando um circuito elementar denominado processo ordinário cíclico ou processo simples cíclico. A seção 4.1 mostrou alguns exemplos de C-lugares. O tempo de duração associado a um C-lugar deve ser sempre nulo e sua marcação inicial deve ser igual ou maior que 1. C-lugares podem ser unidos a processos simples (definição 2.22), ordinários (definição 2.26), com sincronismo (definição 2.23) e com alternância (definição 2.24).

A classificação de lugares adotada neste trabalho inclui todos os processos definidos na seção 2.4 e é diferente da encontrada em Zhou & DiCesare (1993), pois no trabalho de Zhou & DiCesare (1993), os B-lugares são associados a recursos fixos (p.ex.: máquinas e operários) e os C-lugares são associados a recursos variáveis (p.ex.: matéria-prima).

Os conceitos para a classificação dos lugares serão formalizados a seguir.

Seja $m_0$ a marcação inicial de uma rede de Petri Z, seja $p$ um lugar em Z e seja $d_p$ o tempo de duração de espera associado ao lugar $p$. Pode-se particionar o conjunto $P \in Z$ (conjunto de lugares da rede de Petri Z) em três subconjuntos disjuntos denominados A, B e C, que contenham os A-, B- e C-lugares de P respectivamente, tal que $P = A \cup B \cup C$ e $A \cap B \cap C = \oslash$. Assim os conjuntos serão formados pelos lugares definidos da seguinte forma:

**Definição 4.1** – *Um lugar* $p$ *é denominado* **A-lugar** *(ou lugar de operação ) se sua marcação inicial for nula (*$m_0(p) = 0$*) e* $d_p \geqslant 0$*.*

O conjunto de todos os A-lugares de uma rede de Petri será simbolizado por $P_A$.

**Definição 4.2** – *Um lugar* $b$ *é denominado* **B-lugar** *se* $b$ *é inicialmente marcado com* $m_0(B) \geqslant 1$, $d_b = 0$ *e se* $b$ *representar um recurso.*

O conjunto de todos os B-lugares de uma rede de Petri será simbolizado por $P_B$.

**Definição 4.3** - *Um lugar* $c$ *é denominado* **C-lugar** *se* $c$ *é inicialmente marcado com* $m_0(p) \geqslant 1$, $d_c = 0$ *e se* $c$ *representar a peridiocidade de um processo simples.*

O conjunto de todos os C-lugares de uma rede de Petri será simbolizado por $P_C$.

Um A-lugar representa uma tarefa com sua marcação inicial sendo nula, porque inicialmente o sistema deve estar operacional, mas não estar operando.

O conjunto das operações que utilizam o recurso $b$ será simbolizado por $\mathbb{O}_B$. Toda operação que pertence a este conjunto é representada por um lugar $p$ que tenha uma das duas formas mostradas a seguir:

1. $p$ tem como transição de entrada uma das transições de saída de $b$ e como transição de saída uma das transições de entrada de $b$;
2. ou $p$ pertence a um subprocesso (parte de um processo) tal que a operação inicial do subprocesso tem como transição de entrada uma das transições de saída de $b$ e a operação final tem como transição de saída uma das transições de entrada de $b$.

Um B-lugar $b$ pode representar a disponibilidade de um recurso compartilhado por operações, por exemplo, máquinas que executam operações de diferentes processos, ou pode representar a disponibilidade de um recurso dedicado a apenas uma única operação ou a apenas um subprocesso. Com relação à marcação inicial dos B-lugares, um B-lugar $b$ representa a disponibilidade de um recurso simples quando sua marcação inicial é dada por $m_0(b) = 1$, e $b$ representa recursos paralelos, como por exemplo máquinas paralelas idênticas, quando sua marcação inicial é dada por $m_0(b) > 1$.

O conjunto dos processos simples que utilizam o recurso $C$ é simbolizado por $\mathbb{J}_C$. Toda operação inicial de um processo simples que pertence a este conjunto tem como transição de entrada uma das transições de saída de $c$ e toda operação final tem como transição de saída uma das transições de entrada de $c$.

A quantidade de instâncias de um mesmo processo simples que podem ser realizadas ao mesmo tempo, simbolizada por um C-lugar $c$, pode ser chamada de altura de recorrência do processo. A marcação inicial $m_0(c) = n$ significa que quando o sistema estiver sendo operado regularmente, poderão existir $n$ instâncias do processo associado a $c$ sendo realizadas ao mesmo tempo.

Ressalta-se que os B- e C-lugares devem ser definidos *a priori*, por exemplo, pelos projetistas da rede de Petri.

### 4.2.2 Outras Definições e Identificação de Processos

Alguns conceitos propostos por Zhou & DiCesare (1993) serão acrescentados às definições e propriedades de rede de Petri discutidas anteriormente.

Dada uma rede de Petri de sistemas a eventos discretos $Z=< P, T, I, O, D, m_0 >$, um nó é um lugar pertencente ao conjunto P com $P = P_A \cup P_B \cup P_C$ ou uma transição pertencente ao conjunto T. Um **A-caminho** entre $x$ e $y$ é definido como um caminho elementar $EP(x, y)$ (ver seção 3.4), com cada nó sendo um A-lugar ou uma transição, exceto $x$ e $y$.

Um **A-caminho** iniciando e terminando em transições contêm somente lugares inicialmente não marcados (graças a definição de A-lugar). No contexto de manufatura, ele pode representar uma série de operações que podem ser requeridas para finalizar alguns tipos de processos simples ou *jobs*. Por isso, um **A-caminho** pode ser denominado $job$ (de acordo com a definição 2.10) ou processo simples.

O circuito elementar de um processo simples é denominado processo simples cíclico. Quando o processo simples for um processo ordinário (definição 2.26), o circuito elementar será denominado processo ordinário cíclico.

Utilizando-se as definições de circuitos elementares e de A-caminho, define-se:

- Um A-lugar pode pertencer a mais de um A-caminho e também pode pertencer ao conjunto de lugares associados a um B-lugar.
- Um B-circuito é um circuito elementar $EP(x, y)$, com cada nó sendo um A-lugar ou uma transição, exceto $x$ e $y$, com $x = y$ e representado um B-lugar. Processos ordinários cíclicos ou processos simples cíclicos podem ser representados por B-circuitos.
- Um C-circuito é um circuito elementar $EP(x, y)$, com cada nó sendo um A-lugar ou uma transição, exceto $x$ e $y$, com $x = y$ e representado um C-lugar. Processos ordinários cíclicos ou processos simples cíclicos podem ser representados por C-circuitos.
- Uma B-junção é a união de todos os B-circuitos de um mesmo B-lugar. Processos com compartilhamento são representados por B-junções.
- Uma C-junção é a união de todos os C-circuitos de um mesmo C-lugar. Processos com sincronismo ou processos com alternância são representados por C-junções.

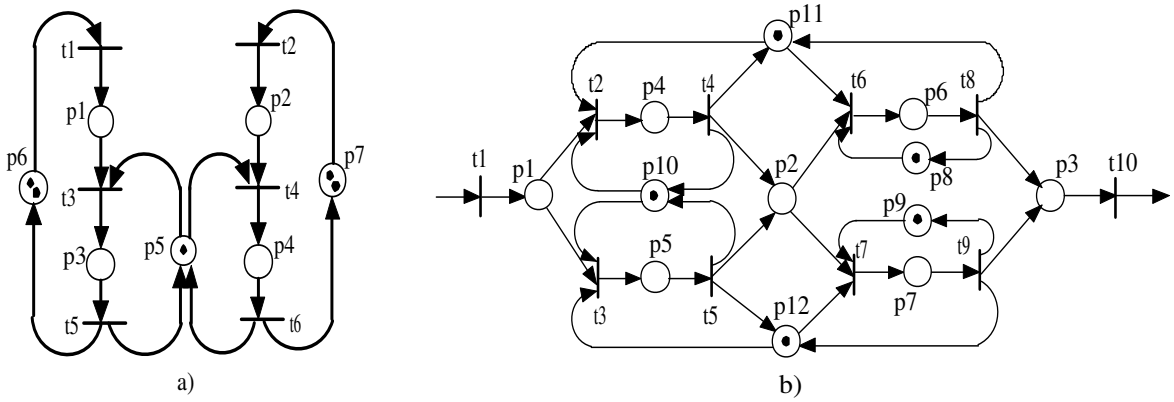



**Figura 4.5**: Modelos de Manufatura em Rede de Petri

A figura 4.5a exemplifica um sistema com 2 C-circuitos e 1 B-junção. Os A-caminhos dos C-circuitos são formados por:

1. $t_1$, $p_1$, $t_3$, $p_3$ e $t_5$;
2. $t_2$, $p_2$, $t_4$, $p_4$ e $t_6$;

Os lugares $p_6$ e $p_7$ são os C-lugares destes C-circuitos, respectivamente.

A B-junção é formada pelos B-circuitos:

1. $t_3$, $p_3$, $t_5$ e $p_5$;
2. $t_4$, $p_4$, $t_6$ e $p_5$;

com o lugar $p_5$ sendo o B-lugar desta B-junção.

Ainda, na figura 4.5a duas características da RdP são evidenciadas:

1. Se o recurso $p_5$ não for considerado, o sistema irá conter 2 processos simples cíclicos independentes, cada qual podendo ser repetido infinitamente.
2. Se $p_5$ for considerado, os dois processos não poderão ser executados simultaneamente. Contudo, no estado inicial ambos já estão prontos para serem executados.

Com relação ao sistema mostrado pela figura 4.5b, quando classificam-se seus lugares em A- B- e C- lugares tem-se:

- **A-lugares:** $p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$ e $p_7$.
- **B-lugares:** $p_8$, $p_9$, $p_{10}$, $p_{11}$ e $p_{12}$.
- **C-lugares:** não existem.

| Elemento | Características |
|---|---|
| A-lugar | é um lugar com marcação inicial nula ($m_0 = 0$). |
| B-lugar | é um lugar com marcação inicial estritamente positiva ($m_0 > 0$) representando um recurso compartilhado ou dedicado. |
| C-lugar | é um lugar com marcação inicial estritamente positiva ($m_0 > 0$) representando a quantidade de instâncias de um mesmo processo simples que podem ser realizadas ao mesmo tempo. |
| A-caminho | é um caminho elementar $EP(x, y)$, com cada nó sendo um A-lugar ou uma transição, exceto $x$ e $y$. |
| B-circuito | é um circuito elementar $EP(x, y)$, com cada nó sendo um A-lugar ou uma transição, exceto $x$ e $y$, e com $x = y$ representado um B-lugar. |
| C-circuito | é um circuito elementar $EP(x, y)$, com cada nó sendo um A-lugar ou uma transição, exceto $x$ e $y$, e com $x = y$ representado um C-lugar. |
| B-junção | é a união de todos os B-circuitos de um mesmo B-lugar. |
| C-junção | é a união de todos os C-circuitos de um mesmo C-lugar. |
| Capacidade de fichas entre $t$ e $t'$ | é o número máximo de disparos da transição $t$ sem disparar a transição $t'$ |

**Tabela 4.1**: Principais conceitos

Percebe-se que a capacidade de processamento dos A-lugares $p_6$ e $p_7$ deste sistema, modelados pelos B-lugares $p_8$ e $p_9$ influencia o comportamento do sistema. Esta influência pode ser medida usando-se a seguinte definição (ZHOU & DICESARE, 1993):

**Definição 4.4** *Dada uma rede marcada* $Z =< P_A \cup P_B \cup P_C, T, I, O, D, m_0 >$*, a capacidade de fichas* $C(t, t')$ *entre as transições* $t$ *e* $t'$ *em* $Z$ *é o número máximo de disparos de* $t$ *sem disparar* $t'$*.*

O valor de $C(t, t')$ depende tanto da marcação inicial quanto da estrutura da rede. Para o sistema mostrado na figura 4.5a, se $m_0 = (0, 0, 0, 0, 1, m_0(p_6), m_0(p_7))^t$, as capacidades de fichas da transição $t_3$ com relação às transições $t_4$ e $t_5$ são:

$$\begin{aligned} C(t_3, t_5) &= \min\{m_0(p_1), m_0(p_5)\}; \\ C(t_3, t_4) &= \begin{cases} 0 & \text{se } m_0(p_1) = 0 \text{ ou } m_0(p_5) = 0; \\ \infty & \text{se } m_0(p_1) > 0 \text{ e } m_0(p_5) > 0. \end{cases} \end{aligned}$$

A tabela 4.2.2 resume as principais definições mostradas nesta seção.

## 4.3 Modelagem de SEDs Periódicos via MILP

Os sistemas a eventos discretos periódicos modelados por redes de Petri, podem ser analisados sob o ponto de vista da Pesquisa Operacional, através do problema de escalonamento cíclico (*Cyclic Scheduling*), com o objetivo de encontrar o melhor funcionamento em regime periódico para o sistema, estabelecendo os inícios de processamento das operações com o menor período possível (ciclo ótimo), respeitando-se as restrições do sistema dadas pela topologia da rede, como será visto nesta seção.

Qualquer operação do conjunto de operações de um Problema de Escalonamento Cíclico (PEC) tem como principal característica a de ser processada infinitas vezes e neste trabalho esta característica é mantida. Assume-se que os atributos das operações incluem um tempo de processamento fixo ($d_i$), uma possível associação a uma ou mais classes de recursos ($m_i$) (sendo possível que as classes de recursos contenham um único recurso), uma vez que a operação começou a ser processada não pode ser interrompida para execução de outra operação, e ainda, as operações são não reentrantes (As operações não se sobrepõem). A k-ésima inicialização e execução de todas as operações de um processo simples será tratada aqui como a k-ésima iteração do processo simples. O tempo transcorrido entre a inicialização da primeira operação de um processo simples, em alguma iteração, e o término da execução da última operação do mesmo processo simples e na mesma iteração será chamado de comprimento do processo simples. O comprimento mínimo de um processo simples será dado pela soma dos tempos de processamento de todas as operações deste processo simples.

Seja $t_{i,k}$ o instante de início da i-ésima operação pela k-ésima vez. É usual em sistemas escalonados ciclicamente, a limitação do número de ciclos iniciados antes do término de algum ciclo iniciado anteriormente (HANEN, 1994; PORTUGAL, 1999; BRUCKER & KAMPMEYER, 2005). Para isto define-se um parâmetro fixo $H \in \mathbb{Z}$, tal que para quaisquer operações $i$ e $j$:

$$t_{j,k+H} \geqslant t_{i,k} + d_i, \quad H \in \mathbb{Z}$$

sendo $d_i$ o tempo de processamento da i-ésima operação.

O parâmetro H é denominado *altura de recorrência* e esta inequação define que a h-ésima recorrência da k-ésima ocorrência da operação $j$ só pode ser inicializada após o término da k-ésima ocorrência de $i$.

Pode-se observar que, em redes de Petri, a altura de recorrência entre operações tem uma correspondência direta com a capacidade de fichas (ver definição 4.4) entre transições. Porém, a altura de recorrência é definida a partir de operações que tenham alguma relação de precedência, o mesmo não ocorrendo com a capacidade de fichas.

Neste trabalho, duas referências básicas para a modelagem de escalonamento cíclico são os modelos propostos por Rao (1992) e por Hanen (1994) mostrados no apêndice B, pois ambos apresentam modelos em programação matemática para problemas de escalonamento cíclico muito sememelhantes aos que são tratados aqui neste trabalho. Hanen (1994) em seu estudo sobre escalonamento cíclico, propôs uma formulação em programação inteira mista para o problema de escalonamento cíclico com recursos disjuntivos, onde qualquer seqüência de processamento das tarefas associadas a um recurso sendo factível, e que denominou job-shop recorrente (JSR). Também Rao (1992) estudou o problema de escalonamento cíclico e propôs uma formulação em programação linear inteira mista (*mixed integer linear programming* – MILP) para o problema de $jobs$ idênticos visando resolver um problema em um ambiente de manufatura, onde um Conjunto de Peças Minimal (MPS) de $jobs$ é processado infinitas vezes. Em ambos, o objetivo é achar um escalonamento periódico com um fluxo máximo, o que equivale a achar um escalonamento periódico com um período mínimo. Tanto em Rao (1992) quanto em Hanen (1994) os tempos de processamento das operações são fixos, não existindo sobreposição de operações em uma mesma máquina e também cada operação é associada a uma única máquina.

Na próxima seção será proposta uma formulação que irá utilizar algumas idéias das formulações de Hanen (1994) e Rao (1992), mostradas no Apêndice B. Estas formulações trataram do problema de escalonamento cíclico de $jobs$ idênticos e máquinas simples. A nova formulação irá tratar do problema mais geral que inclui recursos paralelos e operações sequenciais (definidas na seção 4.3.2) e ainda, processos compostos por sincronismo e por alternância.

### 4.3.1 Simbologia e Variáveis de Decisão

**Os seguintes símbolos serão utilizados nesta seção:**

$\mathbb{J}$ : conjunto de $jobs$ (processos simples);

$J, Q$ : índices relativos a $jobs$ $\quad 1 \leqslant J, Q \leqslant card(\mathbb{J})$;

$\mathbb{M}$ : conjunto de processadores (recursos);

$M, S$ : índice de processadores $\quad 1 \leqslant M, S \leqslant card(\mathbb{M})$;

$\mathbb{O}$ : conjunto de operações;

$\mathbb{O}_M$ : conjunto de operações associadas a um processador;

$\mathbb{O}_M^b$ : conjunto de operações associadas ao exemplar $b$ do processador $M$;

$\mathbb{O}_J$ : conjunto de operações de um $job$ $J$;

$i, j, n$ : índices de operações $\quad 1 \leqslant i, j, n \leqslant card(\mathbb{O})$;

$\sigma_{ij}$ : j é sucessor imediato da operação i;

$\rho_{ij}$ : operação sequencial de i para j.

**Constante**

$w$ : constante de factibilidade.

$l_{i,j}$ : medida de tempo entre a operação i e a operação j.

$d_i$ : duração associada a operação i.

$H_J$ : número de vezes que o job J pode ser iniciado, sem que haja conclusão de alguma instância do mesmo job.

**Variáveis de Decisão**

$z$ : duração do ciclo;

$\tau$ : fluxo do sistema;

$t_i$ : tempo de início de processamento da operação i;

$T_i$ : resto da divisão de $t_i$ por z, isto é, $T_i = t_i \bmod z$;

$\mu_i$ : $T_i$ / z;

$K(i,j)$ : variável associada ao sequenciamento das operações i e j na máquina correspondente;

$X_{ib}$ : variável associada ao sequenciamento das operações i e j no processador correspondente;

$C_J$ : variável binária associada ao *job* J;

$O_i$ : distância em ciclos (*cycle offset*) entre as operações i e $s_i$.

### 4.3.2 Escalonamento de Sistemas a Eventos Discretos Periódicos

Em geral, no estudo do problema de escalonamento cíclico determina-se que todas as operações de um sistema devam ser executadas em cada ciclo. Porém no caso de processos com alternância, esta exigência torna-se excessiva, pois estes processos modelam a existência de caminhos alternativos. No caso destes processos, a exigência deve ser de que todas as operações de pelo menos um de seus processos simples sejam executadas, sendo que se uma operação de um dos processos simples for executada, então todas as operações deste processo simples devem ser executadas. Já nos casos de processos ordinários e de processos com sincronismo todas as operações destes processos devem ser executadas.

Pelas definições dos processos mostradas na seção 2.4, as operações de processos com compartilhamento podem pertencer aos outros processos também. Desta forma, se uma operação de um

processo com compartilhamento também pertencer a um processo com alternância, esta operação poderá não ser executada, porém em qualquer outro caso a operação deverá ser executada.

O problema de escalonamento cíclico de sistemas a eventos discretos periódicos (ESP) inclui restrições de precedência e de não-sobreposição de operações em recursos. As restrições de precedência entre operações modelam processos simples, processos com sincronismo e processos com alternância, como mostrado pela tabela 4.3.2. Já as restrições de não sobreposição de operações em um mesmo recurso modelam processos com compartilhamento. Nestes processos o que se busca é garantir que as operações não utilizem simultaneamente o mesmo exemplar de um recurso.

| | Processos | | | |
|---|---|---|---|---|
| Restrições | com Sincronismo | com Alternância | Simples | com Compartilhamento |
| Precedência Entre Operações | X | X | X | |
| Não–Sobreposição | | | | X |

**Tabela 4.2**: Restrições que modelam os Processos.

Será adotado neste tese, a representação de tempo de início de operação mostrada em Rao (1992), $T_i = t_i \ mod \ z$[1]. Esta forma de representar o tempo possibilita conhecer os tempos de início das operações com relação a duração de ciclo z e desta maneira é possível saber de imediato se a cada ciclo de tempo z, uma operação será executada antes, depois ou ao mesmo tempo de outra operação.

As restrições do modelo aqui proposto, denominado de escalonamento cíclico de sistemas a eventos discretos periódicos, ou simplesmente de escalonamento de sistemas periódicos (ESP), são descritas a seguir:

#### 4.3.2.1 Restrições de Precedência Tecnológica entre Operações

Os processos simples, os com sincronismo e os com alternância podem ser formulados pelo mesmo conjunto de restrições de precedência, mas para um bom entendimento a apresentação será feita em dois passos, o primeiro tratando da formulação dos processos simples e dos com sincronismo e o segundo passo acrescentando os processos com alternância à formulação:

1. Cada processo simples consiste de um conjunto de operações (assumidas inativas em $t = 0$) que devem ser realizadas em uma ordem pré-estabelecida. A partir desta ordem pode-se sugerir um escalonamento, estabelecendo-se os tempos de inicialização de cada operação,

[1] $T_i$ é o resto da divisão de $t_i$ por z

através do seguinte conjunto de inequações ($\forall i \in \mathbb{O}_J, \quad \forall J \subset \mathbb{J}$):

$$\begin{aligned} T_j + O_i z, &\geqslant T_i + l_{i,j}; \\ 0 \leqslant T_i &< z; \\ O &\in \{0,1,2\}; \end{aligned} \tag{4.1}$$

onde $j = s_i$ é o sucessor da operação $i$, o tempo de processamento da operação $i$ é igual a duração associada à operação $i$, isto é, $l_{i,j} = d_i$ e

$$O_i = \begin{cases} 0 & \text{se } s_i \text{ é inicializado no mesmo ciclo,} \\ & \text{que a operação } i; \\ 1 & \text{se } s_i \text{ é inicializado no ciclo seguinte} \\ & \text{relativo ao início da operação } i; \\ 2 & \text{caso contrário.} \end{cases}$$

A soma da direita na inequação 4.1 representa o término do processamento da operação $Ji$ e a soma da esquerda representa o início do processamento da operação $s_{Ji}$.

Para que cada processo do SEDP opere periodicamente é necessário estabelecer uma relação de pseudo-precedência entre sua última operação e sua primeira operação. Esta relação é estabelecida formalmente através da introdução de um C-lugar na Rede de Petri correspondente conforme discutida na seção 4.1. A marcação inicial deste C-lugar representa o número de vezes que um processo simples pode ser iniciado sem que este mesmo processo seja finalizado. Portanto, a cada processo será associado um número $H_J$ que será chamado de altura de recorrência do processo simples J. As duas relações, precedência e pseudo-precedência de recurso, podem ser representadas pelo seguinte conjunto de inequações ($\forall i \in \mathbb{O}_J, \ \forall J \subset \mathbb{J}$):

$$T_j + O_i z \geqslant T_i + l_{i,j}; \tag{4.2}$$

$$\sum_{\forall i \in J} O_i \geqslant 1; \tag{4.3}$$

$$\sum_{\forall i \in J} O_i \leqslant H_J; \tag{4.4}$$

$$0 \leqslant T < z; \tag{4.5}$$

$$O \in \{0,1,2\}; \tag{4.6}$$

onde $j = s_i$ e $l_{i,j} = d_i$. As inequações 4.3 e 4.4 garantem que o processo J seja iniciado e terminado no mínimo uma vez e no máximo $H_J$ vezes.

Se um processo simples estiver ligado diretamente a mais de um C-lugar, cada operação final associada aos C-lugares relaciona-se a cada operação inicial associada a estes mesmos C-lugares, através da inequação 4.2, um exemplo destas redes será mostrado no capítulo 6 na rede da figura 6.1.

Quando um processo simples for associado a um B-lugar b, que represente um recurso dedicado, também cria-se uma relação de pseudo-precedência entre a primeira operação do processo e a última. As inequações 4.2, 4.3, 4.4 e 4.5 também podem ser usadas para modelar este caso.

Quando um lugar $i$ representar a única operação associada a um recurso que por sua vez é representado por um B-lugar $b$, cria-se uma relação de pseudo-precedência entre a operação representada pelo lugar $i$ e a sua operação sucessora imediata. O conjunto formado pelas inequações 4.2, 4.3, 4.4 e 4.5 também pode ser usado para modelar este caso, mas neste caso a variável $l_{i,j}$ será nula, pois neste caso não é necessário esperar o término do processamento da operação sucessora de $i$, pois a operação sucessora não estará associada a $b$, e ainda, o processo simples $J$ será composto por uma única operação, $i$.

Logo, $l_{i,j}$ pode ser definida da seguinte forma:

$$l_{i,j} = \begin{cases} 0 & \text{se } i \text{ for a única operação associada a um recurso,} \\ d_i & \text{caso contrário.} \end{cases}$$

Como dito anteriormente, para processos que sincronizam processos simples (os processos com sincronismo), a inequação 4.2 também será usada para a determinação do escalonamento destes processos. Porém neste caso é necessário garantir que operações que tenham a mesma operação predecessora (dada pela sincronização) sejam iniciadas ao mesmo tempo. A seguinte equação será incluída para este fim:

$$T_j = T_k,\ j = s_i,\ k = s_i,\ i \in \mathbb{O}_J, \mathbb{O}_Q,\ j \in \mathbb{O}_J,\ k \in \mathbb{O}_Q, j \neq k. \tag{4.7}$$

em outras palavras, a tarefa $i$, constitui uma bifurcação (*fork*), seguida pelas tarefas $j$ e $k$. $\mathbb{O}$ é o conjunto das operações do sistema, $J$ e $Q$ são dois processos simples.

2. No caso dos processos que possibilitam a alternância entre processos simples (processos com alternância), existe a possibilidade de que um destes processos simples não seja realizado. Por isso é necessário acrescentar esta possibilidade à equação 4.2. Isto pode ser feito acrescentando-se uma variável binária $c$ associada ao processo $J$, de seguinte forma:

$$C_J = \begin{cases} 1 & \text{se o processo } J \text{ for realizado,} \\ 0 & \text{caso contrário.} \end{cases}$$

Sendo $\omega$ um número suficientemente grande para garantir a factibilidade do conjunto de restrições, o novo equacionamento será ($\forall i, j \in \mathbb{O}_J, \quad n \in \mathbb{O}_Q, \quad r \in \mathbb{O}_J, \quad \forall J, Q \subset \mathbb{J}$):

$$T_j + O_i z \geqslant T_i + l_{i,j} - \omega(1 - C_J), \quad j = s_i; \tag{4.8}$$

$$T_r + O_n z \geqslant T_n + d_n - \omega(2 - C_J - C_Q),\ \forall \sigma_{nr}; \tag{4.9}$$

$$\sum_{\forall j \subset J} C_j \geqslant 1; \tag{4.10}$$

$$\sum_{\forall i \in \mathbb{O}_J} T_i \leqslant \omega \cdot C_J; \tag{4.11}$$

$$T_j = T_k,\ j = s_i,\ k = s_i,\ i \in \mathbb{O}_J, \mathbb{O}_Q,\ j \in \mathbb{O}_J,\ k \in \mathbb{O}_Q, j \neq k; \tag{4.12}$$

$$\sum_{i=1}^{n} O_i \geqslant 1; \tag{4.13}$$

$$\sum_{i=1}^{n} O_i \leqslant H_J; \tag{4.14}$$

$$0 \leqslant T_i < z; \tag{4.15}$$

$$O \in \{0,1,2\}; \tag{4.16}$$

$$C \in \{0,1\}. \tag{4.17}$$

onde $n$ é a última operação do *job* Q mas tem como operação sucessora $r = \sigma_{nr}$ que é a primeira operação de outro *job*, J.

A inequação 4.10 garante que pelo menos um processo simples de um processo composto seja realizado. A inequação 4.8 garante que se um processo simples J for executado ($C_J = 1$), todas as suas operações obedecerão a ordem de precedência. A inequação 4.9 garante que se um processo simples estiver ligado diretamente a mais de um C-lugar, cada operação final associada aos C-lugares esteja relacionada a cada operação inicial associada a estes mesmos C-lugares.

Pela inequação 4.11 se algum $c_J = 0$, isto é, se o processo não for realizado, a soma dos tempos de início das operações pertencentes ao processo deve ser nula, e dado que todos os tempos de início das operações são sempre não-negativos, tem-se que se $c_J = 0$ então todas as operações associadas ao processo J terão tempos de início nulos, por esta razão, $\omega$ deve ser suficientemente grande para garantir a factibilidade da inequação 4.8 e não prejudicar o cálculo de $z$. Um valor razoável para $\omega$ é a soma de todos os tempos de processamento de todas as operações do sistema, ou o dobro disso.

#### 4.3.2.2 Restrições de Não-Sobreposição de Operações:

Os processos com compartilhamento consistem de processos cujas operações compartilham a utilização de um recurso, como uma máquina.

Se um recurso compartilhado for simples (máquina simples), sua marcação inicial será igual a um. Se este recurso estiver representando uma classe de recursos idênticos (ou máquinas paralelas) sua marcação inicial será maior ou igual a 2. Note que máquinas simples são casos particulares de máquinas paralelas, com a classe de recursos idênticos contendo apenas um elemento.

Uma variável $X_{ib}$ será associada à operação $i$, e a cada exemplar da classe de recursos idênticos compartilhados em que a operação $i$ será executada, como descrito a seguir:

$$X_{ib} = \begin{cases} 1 & \text{a operação } i \text{ será executada na máquina } b, \\ 0 & \text{caso contrário;} \end{cases}$$

Para garantir que esta operação seja executada por uma e somente uma máquina a seguinte restrição será utilizada.

$$\sum_{\forall b \subset B} X_{ib} = 1. \tag{4.18}$$

Note que máquinas simples são um caso particular de máquinas paralelas.

É necessário garantir que as operações executadas em uma mesma máquina não se sobreponham. Estas operações podem pertencer também a outros processos, tanto compartilhados como com sincronismo ou com alternância, por isso estes casos precisam ser tratados pela modelagem. Ainda, uma operação pode representar uma seqüência de operações (um subprocesso) e neste caso, o tempo de duração desta operação *sequencial* será variável, pois depende dos tempos de início de execução e da duração das operações incluídas na seqüência. Para distinguir entre uma operação que represente uma seqüência de operações e uma que represente uma única operação, serão adotadas as seguintes designações:

- operação simples: operação única com tempo de duração fixo;
- operação sequencial: operação que representa uma seqüência de operações, com seu tempo de duração dependendo dos momentos de início de execução e da duração das operações incluídas na seqüência.

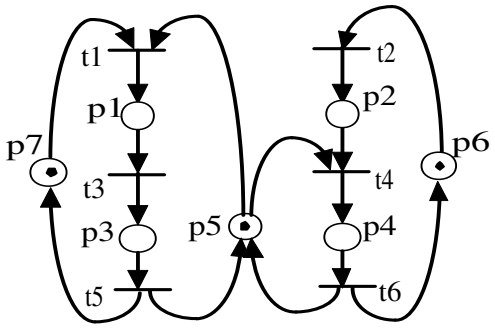



**Figura 4.6**: Processo com Compartilhamento – Operação Sequencial e Operação Simples

Na rede de Petri mostrada pela figura 4.6, os lugares $p_1$ e $p_3$ formam uma operação sequencial no recurso representado pelo lugar $p_5$, enquanto o lugar $p_4$ é uma operação simples para este mesmo recurso.

Note que operações simples são casos particulares de operações sequenciais, com a seqüência de operações sendo composta por uma única operação.

Em um processo com compartilhamento, a máquina à qual todas as operações estão associadas, pode ser simples ou paralela. Logo, quatro combinações entre operações e máquinas são possíveis: operações simples em máquinas simples, operações simples em máquinas paralelas, operações sequenciais em máquinas simples e operações sequenciais em máquinas paralelas. Dentre estas a mais abrangente é a de operações sequenciais em máquinas paralelas, pois abrange todas as outras.

O relacionamento entre estas operações em um processo com compartilhamento pode ser tratado pelo mesmo conjunto de restrições, mas para um bom entendimento, a apresentação destas restrições será feita em três etapas:

- a primeira tratando da formulação dos processos compostos por operações simples e máquinas sequenciais;

- a segunda etapa tratando da modelagem dos processos compostos por operações sequenciais e máquinas simples;

- a terceira acrescentando à formulação o caso de máquinas paralelas.

- **Operações Simples**

  Para garantir que qualquer operação simples que esteja associada à máquina não seja sobreposta por qualquer outra operação as seguintes restrições serão utilizadas ($\forall i, j \in \mathbb{O}_B$, $\forall b \subset B$, $B \subset \mathbb{M}$, $i \in \mathbb{O}_J$ $j \in \mathbb{O}_Q$):

$$
\begin{aligned}
T_i + K_{j,i} z &\geqslant T_j + d_j - \omega(4 - X_{ib} - X_{jb} - C_J - C_Q); && (4.19)\\
T_j + K_{i,j} z &\geqslant T_i + d_i - \omega(4 - X_{ib} - X_{jb} - C_J - C_Q; && (4.20)\\
K_{i,j} + K_{j,i} &= 1; && (4.21)\\
\sum_{\forall b \subset B} X_{ib} &= 1, && (4.22)\\
0 \leqslant T_i &< z, \quad i \in \mathbb{O};\\
K, X, C &\in \{0, 1\};\\
z &\geqslant 0.
\end{aligned}
$$

  As inequações 4.19 e 4.20 buscam o escalonamento das operações nas máquinas levando-se em consideração se as operações estão no mesmo exemplar (variáveis $X_{ib}$ e $X_{jb}$) e se o processo está sendo executado (variáveis $c_J$ e $c_Q$). Já as equações 4.21 e 4.22 garantem a não sobreposição de operações no mesmo exemplar de máquinas. A demonstração desta garantia será feita no final desta seção.

  Note que se $B$ for uma máquina simples todos os $i$ e $j$ vão pertencer à mesma máquina $b$, portanto, $X_{ib} = X_{jb} = 1$.

  A soma da direita na inequação 4.19 representa o término do processamento da operação Qj e a soma da esquerda representa o início do processamento da operação Ji. Da mesma forma, a soma da direita na inequação 4.20 representa o término do processamento da operação Ji e a soma da esquerda representa o início do processamento da operação Qj.

  As variáveis $K_{i,j}$ e $K_{j,i}$ estabelecem dentro de um mesmo período a precedência entre as duas operações $i$ e $j$. É imediato concluir a partir das inequações 4.19, 4.20 e 4.21 que se $X_{ib} + X_{jb} + c_J + c_Q) = 4$ e $K_{i,j} = 0$ então $K_{j,i} = 1$, isto é, operação $i$ precede a operação $j$ em cada ciclo na utilização do recurso.

- **Operações Sequenciais**

  Para uma melhor compreensão das inequações, inicialmente, serão apresentadas as inequações para o caso de operações sequenciais associadas à uma máquina simples, para só depois incluir-se as máquinas paralelas.

  – **Máquinas Simples**

    Seja $\rho_{ab}$ uma operação sequencial com a primeira operação sendo $a$ e a última sendo $b$ e seja $\rho_{cd}$ uma outra operação sequencial definida da mesma forma. Para garantir

que qualquer operação sequencial que esteja associada à máquina não seja sobreposta por qualquer outra operação as seguintes restrições serão utilizadas ($a, b \in \rho_{ab};\ \rho_{ab} \subset \mathbb{O}_J;\ c, d \in \rho_{cd};\ \rho_{cd} \subset \mathbb{O}_Q;\ J, Q \subset \mathbb{J},\ \forall a, b, c, d \in \mathbb{O}_B;\ B \subset \mathbb{M}$):

$$
\begin{aligned}
T_a + K_{ab} z &\geqslant T_b + d_b - \omega(1 - C_J); && (4.23)\\
T_c + K_{cd} z &\geqslant T_d + d_d - \omega(1 - C_J); && (4.24)\\
T_a + K_{ad} z &\geqslant T_d + d_d - \omega(2 - C_J - C_Q); && (4.25)\\
T_c + K_{cb} z &\geqslant T_b + d_b - \omega(2 - C_J - C_Q); && (4.26)\\
K_{ab} + K_{cd} &\geqslant 1; && (4.27)\\
K_{ad} + K_{cb} &\leqslant 1; && (4.28)\\
K_{ab} + K_{cd} - K_{ad} - K_{cb} &= 1; && (4.29)\\
0 \leqslant T_i &< z, \quad i \in \mathbb{O};\\
K, C &\in \{0, 1\};\\
z &\geqslant 0.
\end{aligned}
$$

onde $a, b$ são as operações inicial e final, respectivamente, de uma operação sequencial e da mesma forma $c, d$ são as operações inicial e final de outra operação sequencial.

As inequações 4.23, 4.24 e modelam o relacionamento entre as operações inicial e final de uma mesma operação sequencial ($\rho_{ab}$ e $\rho_{cd}, respectivamente$), com a partícula $\omega(1 - c_J)$ representando a possibilidade destas operações não serem executadas se o processo J não for executado.

As inequações 4.25 e 4.26 modelam o relacionamento entre as operações inicial e final de operações sequenciais diferentes existentes na mesma máquina, com a partícula $\omega(2 - c_J + c_Q)$ representando a possibilidade das operações do processo J ou do processo Q não serem executadas se algum destes processos não forem executados.

As inequações 4.27, 4.28 e 4.29 garantem a não sobreposição de operações na mesma máquina, independentemente das operações serem realizadas ou não.

A figura 4.7 mostra os valores de $K_{ab}$, $K_{cd}$, $K_{ad}$ e $K_{cb}$ nas quatro únicas situações possíveis entre as duas operações sem que haja sobreposição. Já a figura 4.8 mostra os valores para estes K´s quando ocorre sobreposição destas operações.

Pelos valores dos $K's$ mostrados nas figuras 4.7 e 4.8, conclui-se que a soma $K_{ab} + K_{cd}$ sempre é maior ou igual a 1, tanto nos casos de sobreposição quanto nos casos de não sobreposição de operações. A soma $K_{ad} + K_{cb}$ sempre é menor ou igual a 1 no caso de não sobreposição de operações e é sempre maior ou igual a 1 no caso de sobreposição. Estas somas características são representadas pelas inequações 4.27 e 4.28.

Quando não é permitido a sobreposição de operações, existem apenas dois valores possíveis para a soma dos $K's$, $K_{ab} + K_{cd} + K_{ad} + K_{cb} = 1$ ou $K_{ab} + K_{cd} + K_{ad} + K_{cb} = 3$. Já analisando todos os casos em que seja possível existir sobreposição das operações, esta soma resultará sempre em 2 ou 4. Por isto para garantir que não haja sobreposição de operações a soma dos $K's$ deve sempre resultar em um dentre estes dois valores: 1 ou 3. Para representar esta soma usa-se a equação 4.29.

– **Máquinas Paralelas**

No caso de máquinas paralelas apenas quando duas operações estão associadas ao mesmo exemplar de uma classe de máquinas idênticas, é que elas não podem ser sobrepostas, se elas estiverem em exemplares diferentes elas podem ser feitas ao mesmo tempo. Para modelar esta característica, pode-se modificar as inequações 4.25 e 4.26, da seguinte forma ($a, b \in \rho_{ab};\ \rho_{ab} \subset \mathbb{O}_J;\ c, d \in \rho_{cd};\ \rho_{cd} \subset \mathbb{O}_Q;\ J, Q \subset \mathbb{J};\ \forall a, b, c, d \in \mathbb{O}_B;\ \forall m \subset B;\ B \subset \mathbb{M}$):

$$T_a + K_{ab}z \geqslant T_b + d_b - \omega(1 - C_J); \tag{4.30}$$

$$T_c + K_{cd}z \geqslant T_d + d_d - \omega(1 - C_Q); \tag{4.31}$$

$$T_a + K_{ad}z \geqslant T_d + d_d - \omega(4 - X_{am} - X_{cm} - C_J - C_Q); \tag{4.32}$$

$$T_c + K_{cb}z \geqslant T_b + d_b - \omega(4 - X_{am} - X_{cm} - C_J - C_Q); \tag{4.33}$$

$$\sum_{\forall m \subset B} X_{am} = 1; \tag{4.34}$$

$$\sum_{\forall m \subset B} X_{cm} = 1; \tag{4.35}$$

$$K_{ab} + K_{cd} \geqslant 1; \tag{4.36}$$

$$K_{ad} + K_{cb} \leqslant 1; \tag{4.37}$$

$$1 = K_{ab} + K_{cd} - K_{ad} - K_{cb}; \tag{4.38}$$

$$0 \leqslant T_i < z, \quad i \in \mathbb{O};$$

$$K, C \in \{0, 1\};$$

$$z > 0.$$

Se $a$ e $c$ estiverem no mesmo exemplar da máquina, as inequações 4.32 e 4.33 serão equivalentes às inequações 4.25 e 4.26.

A inequação 4.34 garante que a operação $a$ seja executada por apenas um exemplar da classe de máquinas paralelas mesmo que o processo a que ela pertença não esteja sendo

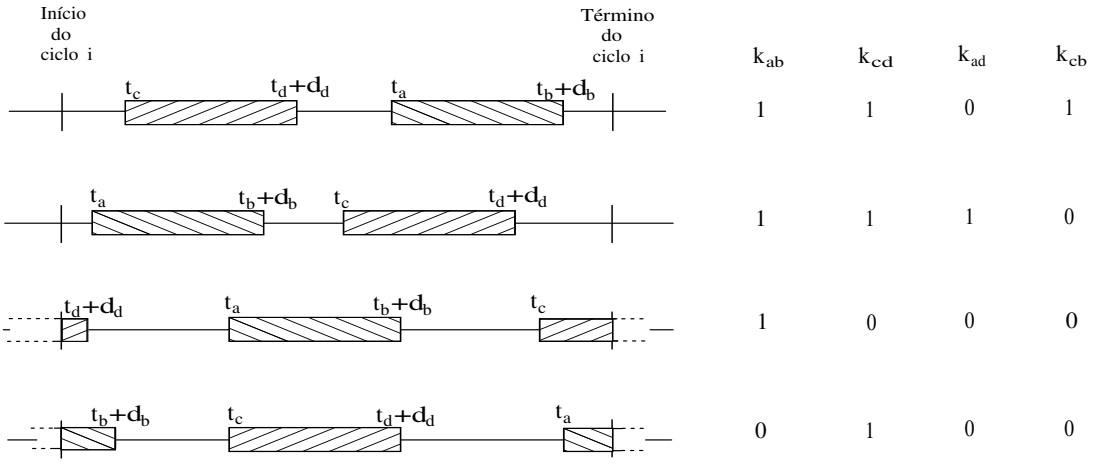



**Figura 4.7**: Seqüências de operações

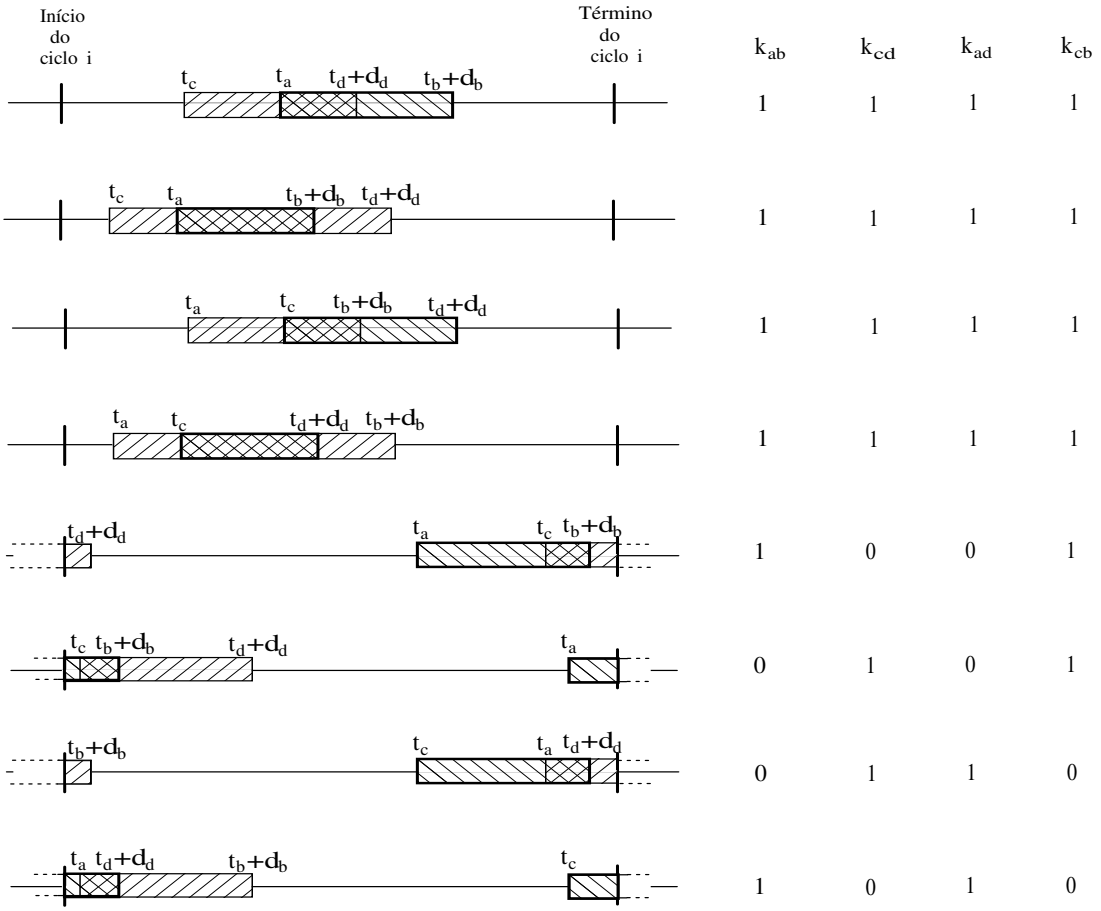



**Figura 4.8**: Sobreposição de Operações Sequenciais

executado. Da mesma forma, a inequação 4.35 garante que a operação c seja executada por apenas um exemplar da classe de máquinas paralelas.

Note que no caso de operações simples, $a = b$ e $c = d$ e o conjunto de inequações fica ($\forall a, c \in \mathbb{O}_B$; $B \subset \mathbb{M}$; $a \in \rho_a$; $\rho_{aa} \subset \mathbb{O}_J$; $c \in \rho_{cc}$; $\rho_{cc} \subset \mathbb{O}_Q$, $\forall m \subset B$):

$$T_a - K_{aa} z \geqslant T_a + d_a - \omega(1 - C_J); \tag{4.39}$$

$$T_c - K_{cc} z \geqslant T_c + d_c - \omega(1 - C_Q); \tag{4.40}$$

$$T_a - K_{ac} z \geqslant T_c + d_c - \omega(4 - X_{am} - X_{cm} - C_J - C_Q)); \tag{4.41}$$

$$T_c - K_{ca} z \geqslant T_a + d_a - \omega(4 - X_{am} - X_{cm} - C_J - C_Q); \tag{4.42}$$

$$\sum_{\forall m \subset B} X_{am} = 1, \tag{4.43}$$

$$\sum_{\forall m \subset B} X_{cm} = 1, \tag{4.44}$$

$$K_{aa} + K_{cc} \geqslant 1; \tag{4.45}$$

$$K_{ac} + K_{ca} \leqslant 1; \tag{4.46}$$

$$K_{aa} + K_{cc} - K_{ac} - K_{ca} = 1; \tag{4.47}$$

$$
\begin{aligned}
0 \leqslant T_i &< z, \quad i \in \mathbb{O};\\
K, X, C &\in \{0,1\};\\
z &> 0.
\end{aligned}
$$

Pelas inequações 4.39 e 4.40, $K_{aa} = 1$ e $K_{cc} = 1$ quando os *jobs* J e Q são executados. Supondo-se que os *jobs* J e Q são executados e manipulando-se as inequações 4.45, 4.46 e 4.47 chega-se a:

$$K_{ac} + K_{ca} = 1$$

que é equivalente a equação 4.21.
Se algum dos processos simples J e Q não for executado, os K´s poderão assumir qualquer valor e as inequações 4.45, 4.46 e 4.47 serão redundantes.

Portanto, as inequações 4.39, 4.40, 4.32, 4.33, 4.34, 4.35, 4.36, 4.37 e 4.38 representam também os casos envolvendo as operações simples.

#### 4.3.2.3 Formulação do Escalonamento de Sistemas Periódicos

Juntando-se as restrições de precedência tecnológica e de não sobreposição de operações na mesma máquina o conjunto de restrições do ESP será ($\forall i, j \in \mathbb{O}_J, r \in \mathbb{O}_J, n \in \mathbb{O}_Q, \rho_{ij} \subset J$, $\rho_{fg} \subset Q, \forall J, Q \subset \mathbb{J}, \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B, \forall b \subset B, B \subset \mathbb{M}$):

$$
\begin{aligned}
min \quad & Z && (4.48)\\
T_j + O_i z &\geqslant T_i + l_{i,j} - \omega(1 - C_J), \quad j = s_i; && (4.49)\\
T_r + O_n z &\geqslant T_n + d_n - \omega(2 - C_J - C_Q), \ \forall \sigma_{nr}; && (4.50)\\
T_j &= T_k, \ j = s_i, \ k = s_i, \ i \in \mathbb{O}_J, \mathbb{O}_Q, \ j \in \mathbb{O}_J, \ k \in \mathbb{O}_Q, j \neq k; && (4.51)\\
T_i + K_{ij} z &\geqslant T_j + d_j - \omega(1 - C_J), \quad \forall i, j \in \rho_{ij}; && (4.52)\\
T_i + K_{ig} z &\geqslant T_g + d_g - \omega(4 - X_{ab} - X_{cb} - C_J - C_Q), \ i \in \rho_{ij}, g \in \rho_{fg}; && (4.53)\\
1 &= \sum_{\forall b \subset B} X_{ib}, \quad \forall i \in \mathbb{O}_B; && (4.54)\\
1 &\leqslant K_{ij} + K_{fg}, \quad \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; && (4.55)\\
1 &\geqslant K_{ig} + K_{fj}, \quad \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; && (4.56)\\
1 &= K_{ij} + K_{fg} - K_{ig} - K_{fj}, \quad \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; && (4.57)\\
1 &\leqslant \sum_{\forall i \in Q} O_i \leqslant H_Q; && (4.58)\\
1 &\leqslant \sum_{\forall j \subset J} C_j; && (4.59)\\
\omega \cdot C_J &\geqslant \sum_{\forall i \in \mathbb{O}_J} T_{Ji}; && (4.60)\\
0 \leqslant T_i &< z, \quad i \in \mathbb{O}; && (4.61)\\
K, X, C &\in \{0,1\}; \quad O \in \{0,1,2\}; && (4.62)\\
z &> 0. && (4.63)
\end{aligned}
$$

As formulações tratadas até aqui tornaram o problema não-linear, para resolver isto, Hanen (1994) sugeriu as seguintes transformações: $\mu_i = t_i/z$ e $\varphi = 1/z$, com $z > 0$. Portanto, para eliminar a não-linearidade dos conjuntos de inequações, divide-se por $z$ os dois lados das inequações 4.49, 4.50, 4.51, 4.52, 4.53, 4.60 e 4.61. Finalmente, a modelagem denominada Escalonamento de Sistemas Periódicos (ESP), será ($\forall i, j \in \mathbb{O}_J, r \in \mathbb{O}_J, n \in \mathbb{O}_Q, \rho_{ij} \subset J$, $\rho_{fg} \subset Q$, $\forall J, Q \subset \mathbb{J}$, $\rho_{ij}, \rho_{fg} \subset \mathbb{O}_B, \forall b \subset B, B \subset \mathbb{M}$):

$$
\begin{aligned}
\max \quad & \varphi && (4.64)\\
\mu_j + O_i &\geqslant \mu_i + \varphi\, l_{i,j} - \upsilon(1 - C_J), \quad j = s_i; && (4.65)\\
\mu_r + O_n &\geqslant \mu_n + \varphi\, d_n - \upsilon(2 - C_J - C_Q),\ \forall \sigma_{nr}; && (4.66)\\
\mu_j &= \mu_k,\ j = s_i,\ k = s_i,\ i \in \mathbb{O}_J, \mathbb{O}_Q,\ j \in \mathbb{O}_J,\ k \in \mathbb{O}_Q, j \neq k; && (4.67)\\
\mu_i + K_{ij} &\geqslant \mu_j + \varphi\, d_j - \upsilon(1 - C_J), \quad \forall i, j \in \rho_{ij}; && (4.68)\\
\mu_i + K_{ig} &\geqslant \mu_g + \varphi\, d_g - \upsilon(4 - X_{ab} - X_{cb} - C_J - C_Q),\ i \in \rho_{ij}, g \in \rho_{fg}; && (4.69)\\
1 &= \sum_{\forall b \subset B} X_{ib}, \quad \forall i \in \mathbb{O}_B; && (4.70)\\
1 &\leqslant K_{ij} + K_{fg}, \quad \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; && (4.71)\\
1 &\geqslant K_{ig} + K_{fj}, \quad \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; && (4.72)\\
1 &= K_{ij} + K_{fg} - K_{ig} - K_{fj}, \quad \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; && (4.73)\\
1 &\leqslant \sum_{\forall i \in Q} O_i \quad \leqslant \quad H_Q; && (4.74)\\
1 &\leqslant \sum_{\forall j \subset J} C_j; && (4.75)\\
\upsilon \cdot C_J &\geqslant \sum_{\forall i \in \mathbb{O}_J} \mu_{Ji}; && (4.76)\\
0 \leqslant \mu_i &< 1, \quad i \in \mathbb{O}; && (4.77)\\
K, X, C \in \{0,1\}; \quad & O \in \{0,1,2\}; && (4.78)\\
\varphi &> 0.
\end{aligned}
$$

### 4.3.3 Considerações a respeito da complexidade da modelagem

Muita pesquisa tem sido feita a respeito da complexidade de problemas em ambientes de manufatura repetitiva.

Este é o caso dos problemas de *flow shop, open shop* e *job shop* cíclicos que podem ser considerados casos particulares de sistemas a eventos discretos periódicos, sendo compostos apenas por processos ordinários e processos com compartilhamento. Para estes problemas Kamoun & Sriskandarajah (1993) mostraram que a busca do melhor escalonamento de *jobs* de problemas de *flow shop, open shop* e *job shop* com máquinas simples em sistemas de manufatura repetitiva é NP-difícil.

No caso do problema de escalonamento cíclico com máquina idênticas e com restrições de precedência, Munier (1996) mostrou que a complexidade deste problema é NP-difícil. Este problema pode ser visto no contexto desta tese da seguinte forma:

- as máquinas idênticas correspondem a B-lugares com duas ou mais fichas na marcação inicial na RPP que modela o sistema a eventos discretos periódicos;
- as operações que utilizam estas máquinas, que são simbolizadas por A-lugares na RPP, e as respectivas máquinas associadas a elas, formam os processos com compartilhamento nesta RPP;
- as restrições de precedência modelam processos ordinários ou processos com sincronismo desta mesma RPP.

Nestes dois casos não são considerados sistemas que possuam processos com alternância. Contudo um sistema contendo processos com alternância pode ser decomposto em processos simples e dado que pelo menos um destes processos simples terá que ser executado, um problema de escalonamento de um sistema composto por processos com compartilhamento e por processos com alternância também será NP-difícil, devido ao que foi demonstrado por Kamoun & Sriskandarajah (1993).

Dado um sistema a eventos discretos periódicos já modelado por uma RPP, podem ser feitas as seguintes considerações com relação à complexidade da formulação (proposta na seção 4.3.2) devido às variáveis inteiras (analisando-se o pior caso para cada variável):

- **variável inteira** $O$

  Para cada operação pertencente a processos ordinários ou a processos com sincronismo ou a processos com alternância está associada uma variável inteira O, que pode assumir os valores 0, 1, 2 (de acordo com a equação 4.1).

  Seja $n$ a quantidade total de operações de um sistema composto por processos ordinários ou a processos com sincronismo ou a processos com alternância, simbolizados por J. A quantidade de combinações possíveis em que todas as variáveis O podem assumir qualquer um dentre os três valores possiveis é $n^3$. Este caso só vai acontecer quando as alturas de recorrências $H_J$´s dos processos J, que limitam os valores de O, forem muito grande (inequação 4.14). Quanto menor a altura de recorrência $H_J$ de um processo J menor a quantidade de combinações das variáveis O associadas ao processo J.

- **variável binária** $C$

  Seja $no$ o número total de operações do sistema (simbolizadas por A-lugares) composto apenas dos processos com alternância e seja $nc$ a quantidade de C-lugares deste sistema. A quantidade máxima em que os processos com alternância podem ser decompostos em processos simples está associada à quantidade de transições de entrada e de saída dos lugares pertencentes a estes processos e é dada pela equação:

  $$m = (SE + SS) - 2 * (no + nc)$$

  onde SE é a quantidade de transições de entrada dos A- e C-lugares, SS é a quantidade de transições de saída destes lugares, e $no + nc$ é a quantidade destes lugares.

A quantidade $m$ simboliza a quantidade máxima de processos simples do sistema e como a variável C está associada aos processos simples, de acordo com a restrição 4.8, a quantidade de variáveis deste tipo será de no máximo $m$ variáveis deste tipo. A quantidade de combinações possíveis em que todas as variáveis C podem assumir qualquer um dentre os dois valores possiveis é $m^2$.

- **variável binária** K

  Seja $nc$ a quantidade de processos com compartilhamento de um sistemas e seja $mc$ a quantidade de operações pertencentes a um processo com compartilhamento, para cada operação pertencente a um processo com compartilhamento estão associadas $mc-1$ variáveis binárias K (equação 4.21). A quantidade de combinações possíveis em que todas as variáveis K podem assumir qualquer um dentre os dois valores possiveis é $\sum_{i=1}^{nc}((mc_i - 1))^2$. Seja Lc, o conjunto dos processos com compartilhamento do sistema, o limite superior para a quantidade máxima de combinações possíveis das variáveis K é dado por

  $$(nc * \max_{i \subset Lc}(mc_i - 1))^2$$

  sendo $\overline{mc-1} = max_{i \subset Lc}\ (mc_i - 1)$, o limite superior para as variáveis K é dado por $(nc * \overline{mc})^2$.

- **variável binária** X

  Sendo $mc$ a quantidade de operações pertencentes a um processo com compartilhamento de $mi$ máquinas idênticas, para cada operação pertencente a um processo estão associadas $mi$ variáveis binárias X, de acordo com a equação 4.18.

  Seja $nc$ a quantidade de processos com compartilhamento de um sistema. A quantidade de combinações possíveis em que todas as variáveis X podem assumir qualquer um dentre os dois valores possiveis é $\sum_{i=1}^{nc}((mc_i * mi_i))^2$. Seja Lc, o conjunto dos processos com compartilhamento do sistema, o limite superior para a quantidade máxima de combinações possíveis das variáveis X é dado por

  $$(nc * \max_{i \subset Lc}(mc_i * mi_i))^2$$

  sendo $\overline{mc * mi} = max_{i \subset Lc}(mc_i * mi_i)$, o limite superior para as variáveis X é dado por $(nc * \overline{mc * mi})^2$.

Considerando um algoritmo de enumeração explícita, tal como o *Branch & Bound*, a quantidade máxima de nós que poderiam ser abertos em um árvore antes de o algoritmo chegar ao valor ótimo é dada pela soma das quantidades estipuladas nos itens anteriores, isto é:

$$n^3 + m^2 + (nc * \overline{mc-1})^2 + (nc * \overline{mc * mi})^2$$

Esta soma é composta por 4 termos sendo os dois primeiros polinomiais, e os dois últimos não polinomiais. Os termos polinomiais são associados a processos ordinários, a processos com sincronismo e a processos com alternância, enquanto os termos não polinomiais estão associados

exclusivamente aos processos com compartilhamento. Desta forma é possível concluir que se um sistema for composto apenas de processos ordinários ou de processos com sincronismo ou de processos com alternância sua complexidade será polinomial, mas qualquer sistema composto por processos com compartilhamento será NP-difícil.

## 4.4 Correspondência entre as Duas Modelagens Estudadas

Baseando-se na definição de redes de Petri de sistemas a eventos discretos periódicos - **RPP** (ver seção 3.6) e na definição do escalonamento de sistemas a eventos discretos periódicos - **ESP** (ver seção 4.3), pode-se concluir que um sistema modelado por uma rede de Petri de Sistemas a Eventos Discretos Periódicos é modelado pelo Escalonamento de Sistemas a Eventos Discretos Periódicos.

De fato, seja N uma rede de Petri de sistemas a eventos discretos periódicos. Suas principais características são:

1. é temporizada com tempo determinístico;
2. seus lugares marcados são não temporizados;
3. é reversível;
4. é consistente;
5. é estruturalmente viva;
6. é conservativa;
7. contém um e um único lugar marcado em cada um de seus componentes conservativos.

Como seus lugares marcados são não temporizados, sabe-se que as condições representadas por estes lugares estão satisfeitas.

Como é temporizada com tempo determinístico, os tempos de duração das atividades são conhecidos, o que possibilita estipular regras para o funcionamento da rede.

Como a rede é reversível, é possível a partir de qualquer escalonamento chegar ao escalonamento ótimo.

Como a rede é consistente é possível estipular um funcionamento repetitivo para ela através de alguma regra que determine os instantes de disparo de suas transições.

Como a rede é estruturalmente viva, existe uma marcação inicial para a qual N é viva e portanto, a partir de qualquer estado alcançável e para qualquer transição t de seu conjunto de transições, existe uma seqüência de disparos que habilita t (conforme a definição da vivacidade de RdP na seção 3.2.1).

Como a rede é conservativa, ela pode ser decomposta em seus componentes conservativos.

Como a rede contém um e um único lugar marcado em cada um de seus componentes conservativos é possível fazer a modelagem em MILP, conforme mostrado na seção 4.3.

Portanto, dada uma rede de Petri de sistemas a eventos discretos periódicos sempre é possível transformá-la em um modelo para o escalonamento de sistemas a eventos discretos periódicos.

## 4.5 Observações Finais

No capítulo 5, será mostrada uma subrotina que traduz o sistema modelado em rede de Petri para a formulação de escalonamento de sistemas periódicos mostrada na seção 4.3. Esta tradução torna possível utilizar o programa comercial **GAMS** para a busca do escalonamento ótimo do sistema a eventos discretos periódicos.

A subrotina inclui regras de redução e de transformação de redes de Petri, técnicas para a verificação das propriedades de redes de Petri de sistemas a eventos discretos periódicos, técnicas para encontrar os componentes conservativos e os componentes repetitivos estacionários das redes, e regras para a tradução das redes na modelagem em MILP.

# 5

# Métodos para a Representação de RPP como Problemas de ESP

No capítulo anterior, ficou estabelecida uma relação entre os tipos de processo considerados neste trabalho (ordinário, com sincronismo, com alternância e com compartilhamento) e o equacionamento MILP que permite a determinação de um escalonamento cíclico.

Entretanto, dado um sistema real e uma representação dele através de uma rede de Petri, nem sempre estão explícitos os processos envolvidos. Desse modo, para que o equacionamento possa ser realizado, duas etapas devem ser previamente consideradas:

- A tradução da RdP dada, em uma forma padrão, que permita a explicitação dos processos (ordinário, com sincronismo, com alternância e com compartilhamento);
- A decomposição da RdP em processos dos tipos considerados e a concomitante verificação das propriedades descritas na seção 3.6.

O propósito deste capítulo é apresentar as técnicas para estas tarefas da seguinte forma. Na seção 5.1 o procedimento geral envolvendo as tarefas é esboçado. Na seção 5.2 são apresentadas as técnicas de tradução de RdP para uma forma padrão, necessária à etapa seguinte, e na seção 5.3 são apresentadas conjuntamente as técnicas de decomposição e verificação. Finalmente, na seção 5.4 os processos e os lugares serão modelados pela formulação MILP.

## 5.1 Procedimento Geral

O procedimento geral para a representação de redes de Petri (associadas a sistemas a eventos discretos periódicos) - RPP, como problemas de escalonamento cíclico - ESP, é feito como descrito a seguir:

1. transformação e/ou redução do tamanho da rede de Petri preservando-se as propriedades de vivacidade e de limitabilidade, visando a redução do tamanho do problema;
2. verificação de que a RdP estudada satisfaz a definição 3.10 de redes de Petri de sistemas a eventos discretos periódicos (ver seção 3.6);
    - verificação das propriedades de consistência e de conservação da rede (ver seção 3.2.3);

- verificação das propriedades de vivacidade e de reversibilidade da rede (ver seções 3.2 e 3.2.3);
- verificação de que cada um de seus componentes conservativos contém apenas um único lugar marcado e de que seus lugares marcados são não temporizados;

3. utilização dos componentes conservativos para classificar os lugares e identificar os processos, e para decompôr os processos com sincronismo, com alternância e com compartilhamento em processos simples (ver seções 4.2.1, 2.4 e 3.5.1);

4. tendo sido definidos os A-, B- e C-lugares e os processos simples, com sincronismo, com alternância e com compartilhamento e verificadas as propriedades, cria-se a formulação em programação linear inteira (ver seção 4.3.2).

As regras de redução ajudam a diminuir o tamanho da rede, expressando as condições de disparo de forma diferente, porém sem modificar seu funcionamento. Já as regras de transformação são usadas para adequar a modelagem em rede de Petri à modelagem em MILP.

As propriedades de conservação e de consistência de redes de Petri são verificadas através da análise de invariância de rede de Petri mostrada na seção 5.3.

Sob um ponto de vista conceitual, o estudo dos componentes conservativos e dos repetitivos estacionários proporciona uma visão decomposta da estrutura da rede de Petri estudada, pois cada componente conservativo é uma sub-rede e a união destes componentes forma a própria rede de Petri, o mesmo ocorrendo nos componentes repetitivos estacionários. Esta visão decomposta será útil para implementar a decomposição das redes de Petri em processos, com cada componente conservativo sendo considerado um processo (ver seção 2.4), possibilitando a montagem da formulação, e sendo também usados para verificar a limitação, a vivacidade e a conservatividade da rede, e com os componentes repetitivos estacionários sendo usados para verificar a repetitividade e a reversibilidade da rede.

Como as redes de Petri de sistemas a eventos discretos periódicos tem algumas propriedades específicas, como por exemplo, a existência de apenas um único lugar marcado em cada um de seus componentes conservativos, deve-se certificar que a RdP estudada possui estas propriedades.

A classificação dos lugares pode ser feita pelo próprio projetista da rede, mas se ele não a fizer, ela será feita de acordo as definições dos lugares dadas na seção 4.2.1. Conhecida a classificação dos lugares, os processos poderão ser identificados. Conhecidos os processos que compõem a rede de Petri, os processos com sincronismo, com alternância e com compartilhamento serão decompostos em processos simples através de um método enumerativo. Esta decomposição é necessária para montar a formulação em MILP.

Se a rede de Petri estudada for uma máquina de estados, ou um grafo de eventos, ou ainda uma rede de escolha livre, pode-se utilizar a análise estrutural dos processos mostrada na seção 3.5.1 para fazer a análise da vivacidade, da limitação e da reversibilidade da rede. Porém se a rede de Petri não corresponder a nenhum destes três casos, a análise de vivacidade e de reversibilidade será feita aproveitando-se os resultados da verificação das propriedades de conservação e de consistência (análise de invariância) e da decomposição dos processos (análise enumerativa) como será visto na seção 5.3.

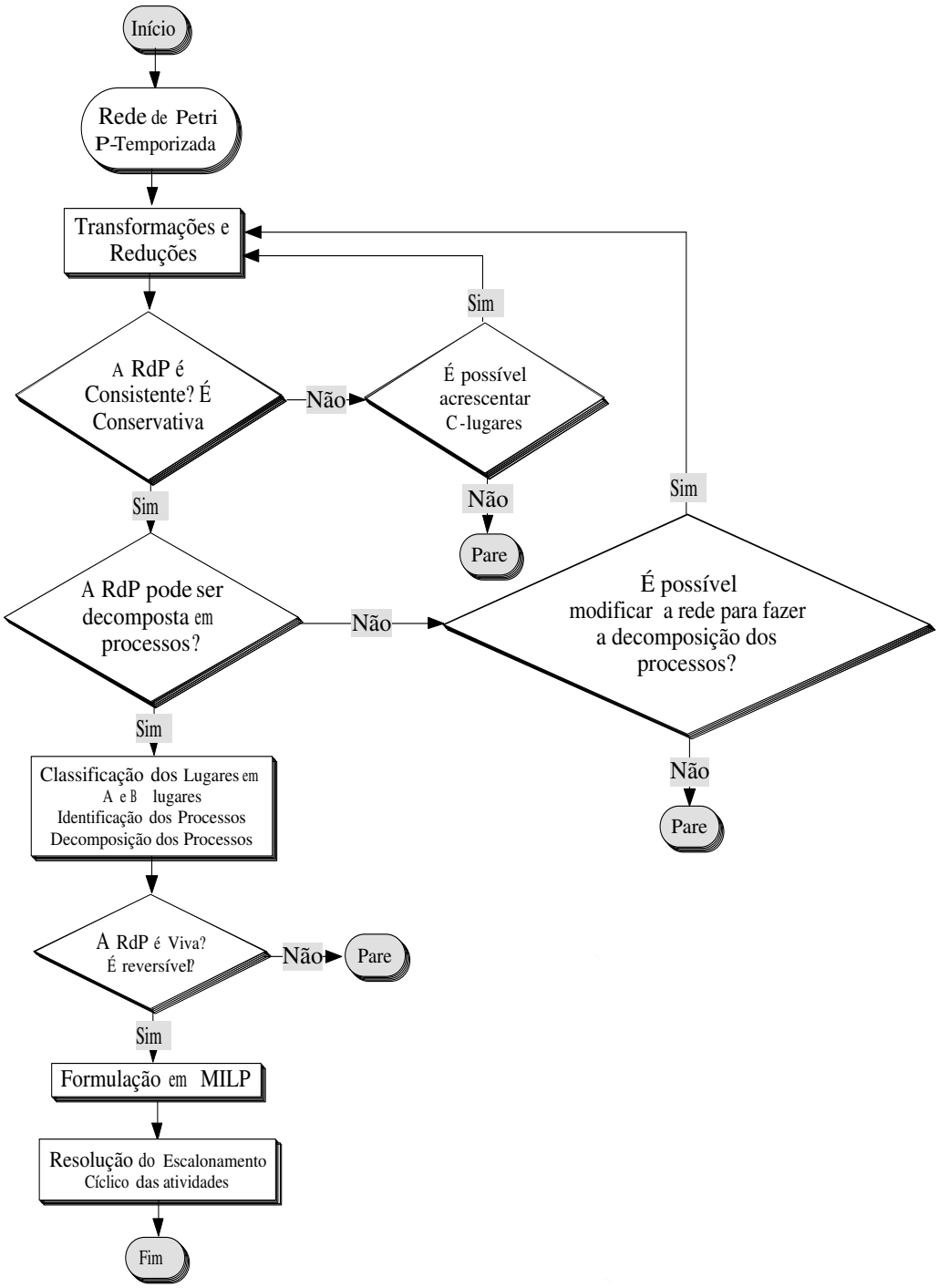



**Figura 5.1**: Fluxograma do algoritmo

A implementação deste procedimento geral será feita pelo Algoritmo Geral, que inclui a tradução da RdP e a otimização do escalonamento do sistema cíclico traduzido, e segue o seguinte roteiro:

1. Classificação da RdP de acordo com a que foi mostrada pela seção 3.1.

2. Se a RdP não for eliminada pela classificação inicial, então deverá passar pelo processo de verificação de propriedades e de decomposição dos processos como mostrado a seguir:
   (a) Redução da RdP, de acordo com as regras que serão mostradas neste capítulo (seção 5.2).
   (b) Verificar a conservação da RdP. Se a rede for conservativa, passa-se para a próxima verificação; se a rede não for conservativa, verificar se é possível torná-la conservativa (acrescentar C-lugares), mas se isto não for possível, não poderá ser traduzida para a modelagem e o algoritmo pára classificando a rede como intratável (seção 5.3.1).
   (c) Verificar a consistência da RdP. Se a rede for consistente, passa-se para a próxima verificação; se for inconsistente a rede não poderá ser traduzida para a modelagem e o algoritmo pára classificando a rede como intratável (seção 5.3.1).
   (d) Verificar se a RdP estudada satisfaz os atributos de redes de Petri de sistemas a eventos discretos periódicos (seção 5.3.2).
   (e) Classificar os lugares da RdP de acordo com o proposto na seção 4.2.1 e identificar os processos de acordo com o proposto na seção 4.2.2(seção 5.3.3).
   (f) Decomposição da RdP em processos (simples, com sincronismo, com alternância e/ou com compartilhamento) (seção 5.3.3).
   (g) Verificar a vivacidade da RdP. Se a rede for viva, passa-se para a próxima verificação; se a rede não for reversível, não poderá ser traduzida para a modelagem e o algoritmo pára classificando a rede como intratável (seção 5.3.4).
3. Resolução do problema de escalonamento de sistemas cíclicos através de algum método exato (por exemplo, *branch & bound*) ou heurístico. Neste trabalho a resolução será feita pelo programa comercial GAMS (Sistema Geral de Modelagem Algébrica) (seção 5.4).

Um fluxograma resumindo este algoritmo é mostrado na figura 5.1.

### 5.1.1 Verificação de Propriedades das Redes de Petri

As técnicas para analisar as propriedades de redes de Petri podem ser divididas em 4 grupos (DiCesare *et. al.*, 1993):

- análise por enumeração – baseia-se na construção da árvore de alcançabilidade da rede (ver seção 3.2). Se rede de Petri é limitada, sua árvore de alcançabilidade é finita e as diferentes propriedades comportamentais podem ser verificadas facilmente, porém se a rede de Petri é ilimitada, sua árvore de alcançabilidade é infinita e portanto impossível de construir. Neste caso, árvores finitas conhecidas como árvores de cobertura podem se construídas. Apesar do seu grande poder de verificação das propriedades comportamentais, a enumeração pode ser difícil de ser aplicada, mesmo em redes com poucos lugares, devido a sua complexidade computacional, pois é fortemente combinatoria.
- análise por transformação – dada uma rede marcada $\langle N, M_0 \rangle$ em que se deseja verificar um conjunto de propriedades $\Phi$, transforma-se esta rede em uma $\langle N', M_0' \rangle$ tal que:

1. $\langle N', M'_0 \rangle$ satisfaça as propriedades $\Phi$ se, e somente se, $\langle N, M_0 \rangle$ as satisfaça. Isto significa que a transformação preservará as propriedades $\Phi$.
2. Seja mais fácil verificar as propriedades $\Phi$ em $\langle N', M'_0 \rangle$ do que em $\langle N, M_0 \rangle$.

Os métodos de redução formam uma classe especial de métodos de transformação nos quais uma seqüência de redes de Petri, preservando as propriedades a serem estudadas, é construída. A construção é feita de tal forma que a rede $\langle N^{i+1}, M_0^{i+1} \rangle$ é *menor* (isto é, tem menos marcações) que a rede anterior na seqüência $\langle N^i, M_0^i \rangle$.

- análise estrutural – esta análise é realizada através do cálculo dos componentes conservativos e repetitivos estacionários e dos invariantes correspondentes (ver seção 3.2).
- análise por simulação – esta análise é feita usando-se uma ferramenta computacional para executar a Rede de Petri. A simulação é uma técnica cara, não exaustiva, e que consome tempo. Ela pode mostrar a presença de propriedades indesejáveis mas não pode provar a correção do modelo no caso geral. Apesar disto, a simulação de RdP é de fato uma abordagem eficaz, direta e conveniente para os engenheiros validarem as propriedades desejadas de um sistema a eventos discretos (WANG, 1998).

Os três primeiros grupos são chamados de métodos estáticos e suas aplicações levam a resultados exatos em redes de Petri como modelos abstratos. Os métodos de simulação são chamados de dinâmicos e *rodam* a rede sobre certas estratégias. Neste caso algumas falhas podem ser detectadas, tais como impasses. Se não forem manifestados problemas durante o processo de simulação, permite-se ter uma certa confiança no modelo. Os métodos de simulação são muito úteis quando o tempo é associado a evolução da rede (redes temporizadas), ou quando deseja-se conhecer a resposta do sistema descrito com a rede em um ambiente que é definido por simulação (DiCesare *et. al.*, 1993).

A análise por simulação não será estudada aqui; para informações mais detalhadas ver Peterson (1981) ou DiCesare *et. al.* (1993). O método utilizado neste trabalho é um método estático posto que serão feitas as análises por transformação (regras de transformação e de redução), estrutural (verificação das propriedades de consistência, de conservação, de vivacidade e de reversibilidade e classificação dos lugares) e por enumeração (decomposição de processos e verificação das propriedades de vivacidade e de reversibilidade).

## 5.2 Regras de Transformação e de Redução

O importante em qualquer processo de transformação é seguir regras que garantidamente preservem as qualidades da RdP, principalmente *vivacidade, limitabilidade e reversibilidade*. A literatura neste assunto é extensa e em particular as transformações relacionadas às temporizações dos componentes da rede são difícieis de serem verificadas sem computador (BERTHELOT, 1985; SLOAN & BUY, 1996; BONHOMME *et. al.*, 1999; BOWDEN, 2000). As transformações de interesse para este trabalho são aquelas que simplificam o processo (regras de redução) ou modificam as temporizações (regras de transformação).

Para que uma transformação possa ser usada é necessário que algumas condições sejam satisfeitas. Estas condições são chamadas de condições de aplicação. Tanto quanto for possível,

estas condições são baseadas na estrutura da rede e são também denominadas condições estruturais. Outras condições de aplicação precisam de conhecimento parcial de seus comportamentos dinâmicos e por isso são denominadas condições comportamentais.

As regras de transformação mostradas a seguir são baseadas nos artigos de Sloan & Buy (1996) e de Bowden (2000). Estes artigos tratam de transformação em redes de Petri temporizadas e por sua vez estão baseados no artigo de Berthelot (1985), que foi o primeiro a estabelecer estas regras de transformação para redes de Petri ordinárias.

Para a auxiliar na formalização destas transformações, serão usados os seguintes símbolos e notações para os conjuntos de predecessores e sucessores de um lugar $p \in P$ e de uma transição $t \in T$ em uma RdP $N =< P, T, I, O, M_0 >$:

- ${}^{o}t$ é o conjunto de todos os lugares de entrada da transição $t$, isto é:

$$ {}^{o}t = \{p \in P | I(p,t) = 1\} \tag{5.1} $$

- $t^{o}$ é o conjunto de todos os lugares de saída da transição $t$, isto é:

$$ t^{o} = \{p \in P | O(p,t) = 1\} \tag{5.2} $$

- ${}^{o}p$ é o conjunto de todas as transições de entrada do lugar $p$, isto é:

$$ {}^{o}p = \{t \in T | I(p,t) = 1\} \tag{5.3} $$

- $p^{o}$ é o conjunto de todas as transições de saída do lugar $p$, isto é:

$$ p^{o} = \{t \in T | O(p,t) = 1\} \tag{5.4} $$

### 5.2.1 Regras para alterações relativas ao tempo

**R1 – Transformação de uma transição temporizada em um lugar temporizado**

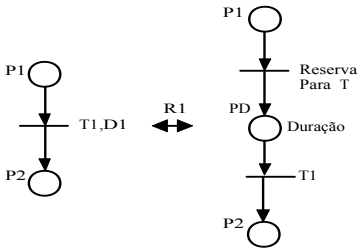



**Figura 5.2**: R1 – Transformação de uma rede de Petri t-temporizada em uma p-temporizada

Segundo Bowden (2000), pode-se transformar uma transição temporizada em uma seqüência *transição-lugar temporizado-transição* equivalente (fig. 5.2). A primeira transição corresponde ao evento instantâneo de início de processamento da tarefa. O lugar serve para memorizar a tarefa sendo executada em um tempo de duração fixo, e a última transição corresponde ao evento instantâneo de fim de processamento da tarefa.

**R2 – Transformação de um lugar marcado temporizado em um lugar não temporizado**

Conforme discutido na classificação dos lugares feita na seção 4.2.1, neste trabalho todos os lugares marcados devem ser não temporizados e por isso pode-se transformar um lugar marcado temporizado em uma seqüência *lugar temporizado–transição–lugar marcado não temporizado* equivalente, como mostrado pela figura 5.2.

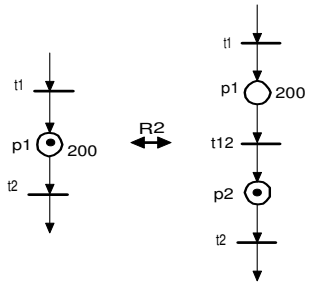



**Figura 5.3**: R2 – Transformação de um lugar marcado temporizado em um lugar não temporizado

## 5.2.2 Regras para a remoção de lugares

As transformações deste tipo não modificam o funcionamento de uma rede, mas as condições de disparo são expressas de outra forma.

**R3 – Fusão de lugares redundantes paralelos**

Se dois lugares tem exatamente as mesmas transições de entrada e de saída e marcações iniciais idênticas, então o lugar com o menor tempo de duração pode ser removido (ver figura 5.4) sem alterar o comportamento da rede (BONHOMME *et. al.*, 1999).

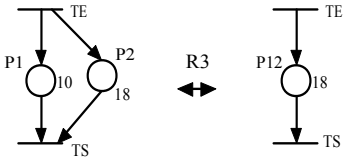



**Figura 5.4**: R3 – Fusão de lugares redundantes paralelos

**R4 – Fusão Serial**

A transformação R4 funde dois lugares $p_1$ e $p_2$ conectados por t que é ao mesmo tempo a única saída de $p_1$ e a única entrada de $p_2$.

**Proposição 5.1** *(Bonhomme et. al., 1999) Seja* $N = (P, T, I, O, D, m_0)$ *uma rede de Petri p-temporizada. Dados dois lugares* $p_1$ *e* $p_2$ *e uma transição* $t_i$ *pertencentes a N, estes lugares podem ser fundidos se satisfizerem as seguintes condições :*

1. *Os lugares* $p_1$ *e* $p_2$ *são inicialmente não marcados.*

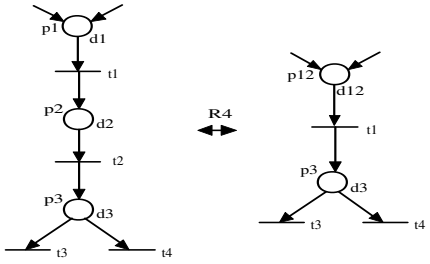



**Figura 5.5**: R4 – Fusão Serial

2. $p_1$ *é a única entrada da transição* $t_i$ *e* $p_2$ *é a única saída desta transição, isto é:*
   ${}^{o}t_i = \{p_1\}$ *e* $t_i^{o} = \{p_2\}$
3. $t_i$ *é a única saída do lugar* $p_1$ *e é a única entrada do lugar* $p_e$*, isto é:*
   ${}^{o}p_1 = \{t_i\}$ *e* $p_2^{o} = \{t_i\}$

*Regra:* Os dois lugares $p_1$ e $p_2$ podem ser substituídos por um único lugar $p_{12}$ tal que:

$$d_{12} = d_1 + d_2$$

$$e \quad \begin{cases} {}^{o}p_{12} = {}^{o}p_1 \\ p_{12}^{o} = p_2^{o} \end{cases} \tag{5.5}$$

**R5 – Remoção de lugares de início vazios**

Se um lugar p não contiver nenhuma transição de entrada, a duração associada a ele for nula e ele for inicialmente não marcado, então ele pode ser removido assim como suas transições de saída (SLOAN & BUY, 1996).

**R6 – Eliminação de Ciclo Próprio**

Um lugar p representa um ciclo próprio em uma rede de Petri N se p possuir exatamente uma transição de entrada e uma transição de saída e elas forem iguais. Um exemplo deste caso é dado pela figura 5.6.

*Regra:* Remover o lugar $p_d$ e todos os seus arcos incidentes também.

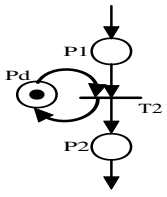



**Figura 5.6**: R6 – Eliminação de Ciclo Próprio

Apesar de ser possível criar-se um algoritmo para fazer estas transformações, existem programas computacionais em que é possível aplicar estas regras de transformação e de redução à rede de Petri modelada, tais como:

- HELENA – High LEvel Net Analyzer, como parte do projeto Quasar desenvolvido pelo instituto Conservatoire National des Arts et Métiers (CNAM) em Paris, França. Disponível, em 30 de outubro de 2006, no seguinte endereço :
  *http://helena.cnam.fr/*.

- INA – Integrated Net Analyzer, desenvolvido pelo Prof. Dr. Peter H. Starke na Universidade Humboldt (Humboldt-Universität) em Berlim, Alemanha. Disponível, em 30 de outubro de 2006, no seguinte endereço :
  *http://www2.informatik.hu-berlin.de/lehrstuehle/automaten/ina/*.

- OPERA – desenvolvido pelo Prof. Dr. Dmitry A. Zaitsev da *Odessa National Telecommunication Academy* em Odessa, Ucrânia. Disponível, em 30 de outubro de 2006, no seguinte endereço :
  *http://http://geocities.com/zsoftua/softe.htm#soft*.

- PEP – Programming Environment based on Petri Nets, desenvolvido inicialmente pelo grupo composto do Prof. Dr. Eike Best, do Dr. Hans Fleischhack e do Dr. Bernd Grahlmann na Universidade de Oldenburg, em Oldenburg, Alemanha. Disponível, em 30 de outubro de 2006, no seguinte endereço :
  *http://parsys.informatik.uni-oldenburg.de/ pep/PEP2.0/index.html*.

## 5.3 Representação de RdP na modelagem de Escalonamento de Sistemas Periódicos

Como visto na seção 4.2.1 os lugares de uma rede de Petri podem ser classificados em função do Problema de Escalonamento de Sistemas Cíclicos da seguinte forma:

- A-lugares: representam tarefas com tempo de duração finito e fixo (conforme ZHOU & DICESARE, 1991);
- B-lugar: representa a disponibilidade de um recurso. O tempo de duração associado a um B-lugar deve ser sempre nulo e sua marcação inicial deve ser igual ou maior a 1.
- C-lugar: representa a periodicidade de um processo simples e une a última tarefa com a primeira tarefa de um mesmo processo ordinário ou de um mesmo processo simples, criando um circuito elementar denominado processo ordinário cíclico ou processo simples cíclico. O tempo de duração associado a um C-lugar deve ser sempre nulo e sua marcação inicial deve ser igual ou maior que 1.

A classificação dos lugares de uma RdP, a decomposição desta rede em processos e a montagem da formulação pode ser feita através dos componentes conservativos e repetitivos da rede como será visto nas próximas seções.

E ainda, se a rede de Petri estudada for uma máquina de estados, ou um grafo de eventos, ou ainda uma rede de escolha livre, usa-se a análise estrutural dos processos mostrada na seção 3.5.1, para fazer a análise da vivacidade, da limitação e da reversibilidade da rede. Caso contrário, esta análise será feita ao mesmo tempo que a decomposição dos processos, mostrada na seção 5.3.3.

A representação de RdP em ESP seguirá os seguintes passos:

- Encontrar os componentes conservativos e os repetitivos estacionários da rede, utilizando o algoritmo proposto na seção 5.3.1.
- Verificar se a RdP estudada satisfaz os atributos de redes de Petri de sistemas a eventos discretos periódicos (seção 5.3.2).
- Classificar os lugares da rede (seção 5.3.3).
- Decompor a rede em processos(seção 5.3.3).
- Verificar as propriedades de limitação, de vivacidade e de reversibilidade da rede (seção 5.3.4).

### 5.3.1 Algoritmo Simplificado para a Análise Estrutural

Na seção 3.2.2 mostrou-se que as propriedades estruturais estão ligadas a topologia da rede, não dependendo de sua marcação inicial da rede. Elas são definidas através dos componentes conservativos de lugar e dos componentes repetitivos estacionários. Nesta seção será apresentado um algoritmo clássico para a obtenção destes componentes, assim como uma versão simplificada para o mesmo algortimo.

Segundo Cardoso & Valette (1997), é possível obter-se informações adicionais sobre o comportamento dinâmico da rede de Petri, a partir destes elementos estruturais e utilizando também a informação sobre a marcação.

O cálculo de p-invariantes minimais e t-invariantes minimais (ver seção 3.2.3) tem sido extensivamente estudado, devido a quantidade exponencial de invariantes que podem ocorrer e a complexidade de tempo que pode ser demandado para calculá-las.

Segundo Colom & Silva (1991), os p-invariantes minimais podem ser calculados eficientemente usando-se técnicas de álgebra linear, pois eles devem satisfazer a seguinte equação:

$$X^t.C = 0 \tag{5.6}$$

sendo X uma matriz cujas colunas são todos os p-invariantes minimais da rede. Esta equação é obtida a partir da equação fundamental (eq. 3.2) com $X_t.m_0 = X_t.m'$ (ver discussão na seção 3.2.2).

Os t-invariantes minimais também podem ser calculados usando-se técnicas de álgebra linear, posto que eles devem satisfazer a seguinte equação:

$$C.Y = 0 \tag{5.7}$$

sendo Y uma matriz cujas colunas são todos os t-invariantes minimais da rede. Esta equação é obtida a partir da equação fundamental (eq. 3.2) com $m_0 = m'$ (ver discussão na seção 3.2.2).

Para resolver estas equações, Colom e Silva (1991) propõem um estudo misturando-se técnicas de geometria convexa com aquelas já desenvolvidas em Redes de Petri. O algoritmo proposto por Colom e Silva (1991) é mostrado a seguir:

**Algoritmo 1:** Obter o p-semifluxo minimal a partir da matriz de fluxo ou matriz de incidência **C** através da eliminação gaussiana:

1. Dada a matriz $C_{n \times m}$

   (a) $A := C$; $F := I_n$ $\quad$ {$I_n$ é uma matriz de identidade de dimensão} $n$

   (b) **Para** $i := 1$ **até** $m$ **faça** $\quad$ {$m$ é o número de transições da rede}

   **Adicionar** à matriz $[A \mid F]$ todas as linhas que são combinações lineares de pares de linhas de $[A \mid F]$ e que anulam a i-ésima coluna de A.

   **Eliminar** da matriz $[A \mid F]$ as linhas em que a i-ésima coluna de A não é nula.

   (c) As linhas da matriz F são os p-semifluxos positivos (p-invariantes) da rede. Entre eles estão todos os p-semifluxos minimais.

As operações efetuadas sobre as linhas de A são memorizadas sob a forma de somas formais. Cada coluna de F é memorizada sob a forma de uma combinação linear de suas colunas originais ($f_i + f_j$ descreve uma coluna obtida através da soma das colunas $i$ e $j$ de $F_0$). A cada eliminação de variável, a coluna de A que serviu para eliminar uma variável é eliminada, assim como a linha que corresponde a esta variável. Como a variável não faz parte da base de soluções, a coluna correspondente de F também será anulada.

Seja $r$ o posto (*rank*) da matriz C, seja $n$ o número de lugares da RdP e seja $\mathbb{P}$ o espaço vetorial dos componentes conservativos que coincide com o espaço nulo (*kernel*) de $C^t$. A dimensão de $\mathbb{P}$ é dada por:

$$dim(\mathbb{P}) = n - r$$

Seja $\mathbb{T}$ o espaço vetorial dos componentes repetitivos estacionários. A dimensão de $\mathbb{T}$ (espaço nulo de C) é dada por:

$$dim(\mathbb{T}) = q - r$$

onde $q$ é o número de transições.

Segundo Cardoso & Valette (1997), no caso dos p-invariantes minimais, apenas o cálculo das colunas da matriz F que correspondem às $n - r$ componentes da base das soluções é útil. De fato, qualquer combinação linear que envolva os mesmos elementos da base terá o mesmo suporte levando portanto ao mesmo p-invariante minimal. Desse modo, todos o p-invariantes minimais podem ser obtidos a partir de todas as combinações (2 a 2, 3 a 3,..., (n-r) a (n-r)) de elementos da base. Por isso, as outras colunas de F podem ser ignoradas, pois são combinações lineares dos componentes da base. Com base nestas observações, Cardoso & Valette (1997) propuseram um algoritmo simplificado contendo 4 etapas:

**Algoritmo Simplificado**

E1 Procurar uma coluna que possua um só termo diferente de zero. Ela corresponde a uma equação da base de equações e é apagada, sendo a variável correspondente ao termo não nulo eliminada, pois como a única solução possível é nula, não pode fazer parte da base de soluções. Apagar a linha de A associada a esta variável e a coluna de F correspondente. Repetir a etapa 1 enquanto for possível, depois ir para a etapa 2.

E2 Procurar uma coluna tendo apenas um componente não nulo com um sinal dado; os outros devem ser nulos ou de sinal contrário (um positivo e todos os outros negativos ou nulos,

por exemplo). A variável correspondendo a esta linha é eliminada efetuando-se apenas combinações positivas de linhas de modo a gerar o aparecimento de zeros na coluna considerada. Após esta operação, eliminar esta coluna, a linha de A que corresponde à variável eliminada e a coluna de F associada. Se tal coluna é encontrada, voltar a etapa 1 senão ir para a 3.

E3 Procurar uma coluna tendo $i \geqslant 2$ componentes não nulos positivos e $j \geqslant 2$ componentes não nulos negativos. Com a ajuda de um dos componentes não nulos positivos, anular $j - 1$ componentes negativos. Voltar a etapa 2 para eliminar a variável que corresponde a um único componente negativo não anulado. É preciso salientar que foi escolhido um componente positivo entre $i$ e um componente negativo entre $j$. Existem $i \times j$ maneiras de proceder. Esta escolha pode ter conseqüência sobre a forma da base obtida. Se nenhuma coluna que corresponda ao critério acima é encontrada, ir para o passo 4.

E4 Se existe uma coluna tendo todos os componentes de mesmo sinal, prosseguir a eliminação de Gauss para esta coluna, porém a base não possuirá mais apenas soluções positivas; em seguida, ir para a etapa 1, senão o algoritmo termina. Ou todas as colunas de A são nulas, ou a matriz não possui mais nenhum elemento. As combinações não apagadas que correspondem à construção de colunas de F fornecem diretamente os vetores da base.

Após a aplicação deste algoritmo, cada linha da matriz F composta por elementos não nulos é denominada por p-semifluxo minimal da rede, sendo que um p-semifluxo positivo minimal é um p-invariante minimal da rede.

O algoritmo foi explicado apenas para o componente conservativo (mas basta calcular a transposta da matriz C para calcular os componentes repetitivos estacionários, usando uma matriz G como matriz auxiliar para memorizar as combinações lineares das colunas de C, da mesma forma que a matriz F memoriza as combinações lineares das colunas de $C^t$).

Ao final da aplicação do algoritmo simplificado na busca dos componentes conservativos, as linhas não negativas e não nulas da matriz F corresponderão aos vetores da base dos componentes conservativos. E ao final da aplicação deste algoritmo na busca dos componentes repetitivos estacionários, as linhas não negativas e não nulas da matriz G corresponderão aos vetores da base dos componentes repetitivos estacionários.

### 5.3.2 Verificação das Propriedades de Sistemas Periódicos

Após aplicar o algoritmo simplificado a uma rede de Petri em busca dos componentes conservativos é necessário analisar a matriz F resultante, para verificar se todos os vetores resultantes correspondem a componentes conservativos de RPP (ver definição 3.10).

Também é necessário após a busca dos componentes repetitivos estacionários de uma RdP, analisar a matriz G resultante do algoritmo simplificado, a fim de verificar se todos os vetores resultantes correspondem a componentes repetitivos estacionários.

Ainda é necessário verificar a partir da matriz de incidência C, da matriz F resultante da análise de p-invariância e da matriz G resultante da análise de t-invariância se algum dos seguintes casos acontece:

Caso 1 – algum componente conservativo não possui lugares marcados;
Caso 2 – alguma linha de F possui elementos negativos;
Caso 3 – algum componente conservativo possui dois ou mais lugares marcados;
Caso 4 – dois lugares ligados pela mesma transição, sendo um predecessor do outro, não fazem parte de um mesmo componente conservativo;
Caso 5 – alguma transição não está incluída em qualquer componente repetitivo estacionário ou alguma linha de G possui elementos negativos.

A ocorrência do caso 1 significa que a rede não será conservativa, do caso 2, que a rede não será viva, do caso 3, que a rede não satisfaz a definição 3.10 de redes de Petri de sistemas a eventos discretos periódicos e dos casos 4 e 5, que a rede não é repetitiva.

Deve-se ressaltar que se algum destes 5 casos ocorrer, a rede de Petri não poderá ser tratada pela nossa modelagem, a menos que esta rede de Petri seja transformada em uma que possa ser tratada. Neste caso, a nova rede de Petri poderá ter um comportamento diferente da rede de Petri anterior.

Deve-se ressaltar ainda que as regras estudadas nesta seção são apenas sugestões para a solução dos problemas encontrados, sendo possível para o projetista da rede resolver estes problemas de outra forma.

Se não houver nenhum problema na matriz F resultante e nem na matriz G resultante, os componentes conservativos poderão ser denominados por processos, a RdP será decomposta nestes processos, e estes processos tornarão possível a formulação do problema de ESP.

Para tratar os casos mencionados anteriormente, as seguintes regras de verificação dos componentes são dadas (a numeração seguirá a das transformações da seção anterior):

**R7** – Busca de precedências conflitantes (Caso 1)

A precedência conflitante acontece quando um p-invariante não contém fichas na marcação inicial, sendo portanto uma seqüência não realizável, conforme discutido na seção 3.2.3.

Para eliminar a precedência conflitante pode-se aplicar uma das duas regras:

- se o componente conservativo contiver um lugar não temporizado, coloca-se uma ficha neste lugar.
- se o componente conservativo for composto apenas de lugares temporizados, deve-se retirar algum destes lugares, mas a forma que esta retirada será feita cabe ao projetista decidir.

Se não for possível acabar com precedência conflitante, a rede de Petri será considerada não tratável e o Algoritmo Geral é finalizado.

**R8** – Simplificação de caminhos redundantes (Caso 2)

Os caminhos redundantes acontecem quando um p-semifluxo contêm algum valor negativo não sendo portanto um componente conservativo.

Para eliminar o caminho redundante a partir dos elementos (lugares) não nulos do p-semifluxo, deve-se verificar se a rede formada por este subconjunto de elementos é conectada ou se ele é fortemente conectada, de acordo com as definições 3.4 e 3.5. Se a rede for apenas conectada a regra R8.1 deverá ser usada e se ela for fortemente conectada usa-se a regra R8.2. Estas regras são descritas a seguir:

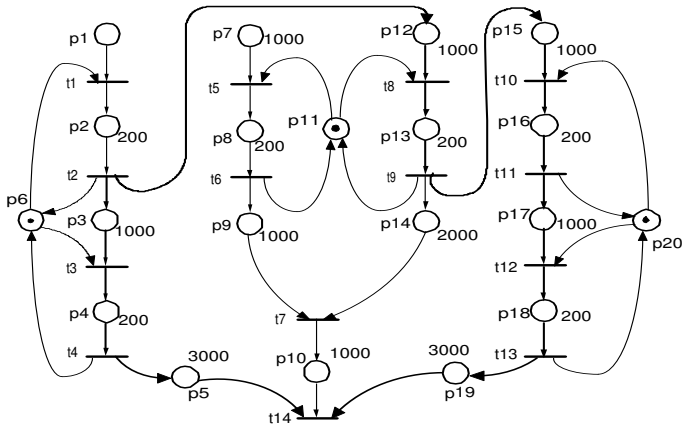



**Figura 5.7**: Rede de Petri com Caminhos Redundantes

R8.1 Se um subconjunto de elementos não nulos do p-semifluxo formarem um caminho, mas não um circuito, pode-se adicionar um C-lugar ligando a última transição do caminho à primeira transição do caminho formado por estes elementos (isto irá criar um circuito). Deve-se fazer isto para todos os caminhos que não pertençam a nenhum circuito, aproveitando-se inclusive os C-lugares que já tiverem sido incluídos.

Como exemplo, considere o problema de escalonamento mostrado pela rede de Petri da figura 5.7. Usando-se o algoritmo simplificado para analisar a p-invariância, encontram-se três p-invariantes, um p-semifluxo com alguns elementos negativos e alguns lugares, como $p_1$ e $p_7$, que não pertencem nem a um p-invariante nem a um p-semifluxo. O p-semifluxo encontrado é gerado pelo conjunto de lugares:

$$\{p_3, p_4, p_5, p_{12}, p_{13}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}\}$$

que é dado por:

$$p_3 + p_4 + p_5 - p_{12} - p_{13} - p_{15} - p_{16} - p_{17} - p_{18} - p_{19} = 0$$

Este p-semifluxo mostra que os lugares $p_3, p_4$ *e* $p_5$ formam um caminho e os lugares $p_{12}, p_{13}, p_{15}, p_{16}, p_{17}, p_{18}$ *e* $p_{19}$ formam outro, sendo que estes dois caminhos são redundantes entre si.

A inclusão de 2 C-lugares, $p_{21}$ *e* $p_{22}$, conforme ilustrado pela figura 5.8, resolvem o problema. Para verificar se a redundância foi removida, aplica-se o algoritmo simplicado à nova rede, que leva aos conjuntos de lugares mostrados a seguir, onde cada conjunto de lugar forma um componente conservativo:

- $\{p_1, p_2, p_3, p_4, p_5, p_{21}\}$
- $\{p_2, p_4, p_6\}$
- $\{p_8, p_{11}, p_{13}\}$
- $\{p_1, p_2, p_{10}, p_{12}, p_{13}, p_{14}, p_{21}\}$
- $\{p_1, p_2, p_{12}, p_{13}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}, p_{21}\}$
- $\{p_{16}, p_{18}, p_{20}\}$
- $\{p_7, p_8, p_9, p_{10}, p_{22}\}$

R8.2 Quando a rede de Petri é fortemente conectada, a existência de um p-semifluxo com valores negativos ao final da aplicação do algoritmo simplificado pode tornar impossível a decomposição desta rede em processos, pois:

- se um lugar estiver associado a um elemento negativo de um p-semifluxo, e não estiver associado a nenhum outro p-semifluxo, então a rede não será uma RPP conservativa e por isso não poderá ser decomposta em processos e;
- se um lugar estiver associado a um elemento negativo de um p-semifluxo, e também estiver associado a um elemento positivo de um outro p-semifluxo, então será possível fazer combinações lineares das linhas da matriz F de forma que este lugar esteja associado a apenas elementos positivos nos p-semifluxos (conseqüência direta do algoritmo simplificado apresentado anteriormente). Assim, dado um subconjunto de lugares não nulos de um p-semifluxo, $\alpha$, contendo elementos negativos, se todos os lugares associados aos elementos negativos de $\alpha$, estiverem associados a elementos positivos em outros p-semifluxos, o p-semifluxo $\alpha$ poderá ser modificado fazendo-se combinações lineares das linhas da matriz F de forma a ter apenas elementos não negativos.

  Neste caso, a rede será conservativa e os elementos não nulos em $\alpha$ formarão um componente conservativo. Porém, algumas linhas de F poderão conter valores maiores que 1 e a rede não poderá ser decomposta em processos. A razão disto é que para que todos os lugares do componente conservativo sejam alcançados, é necessário que algumas transições de entrada destes lugares (as que tem valores superiores a 1), sejam disparadas mais de uma vez, e isto significa que as operações simbolizadas por estes lugares seriam executadas mais de uma vez em um ciclo, porém isto não pode ser modelado na nossa decomposição dos processos. Por esta razão, este funcionamento da rede será considerado aqui neste trabalho como um *mau funcionamento* da rede.

Se a causa do *mau funcionamento* for a presença da estrutura denominada de escolha–

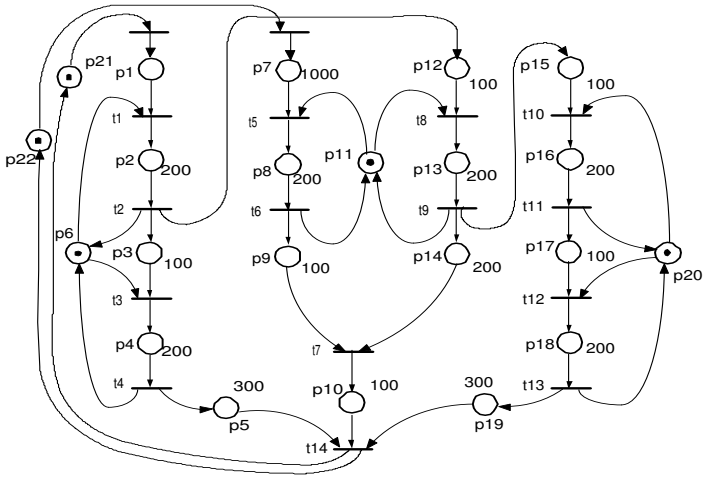



**Figura 5.8**: Rede de Petri Conservativa

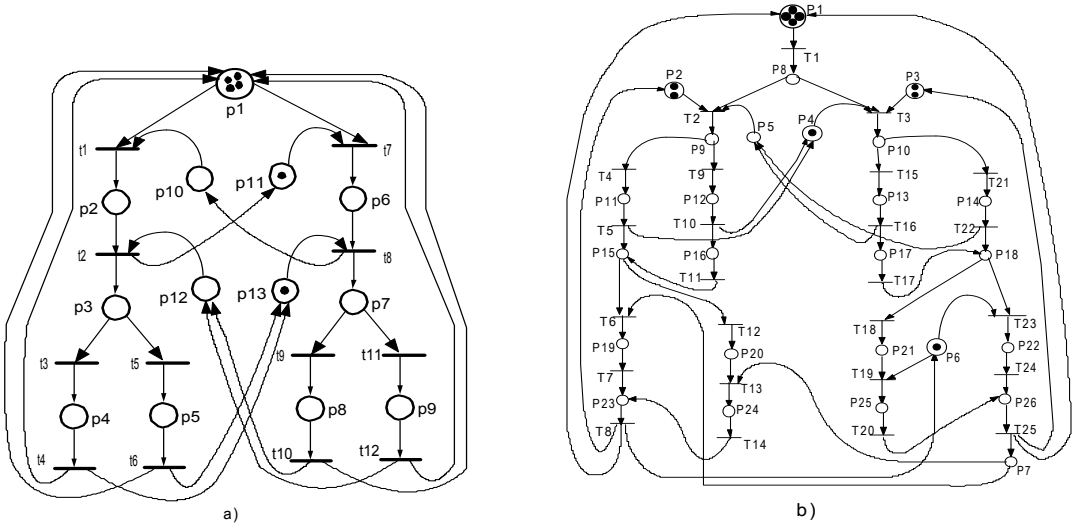



**Figura 5.9**: Exemplos de Escolha Sincronizada

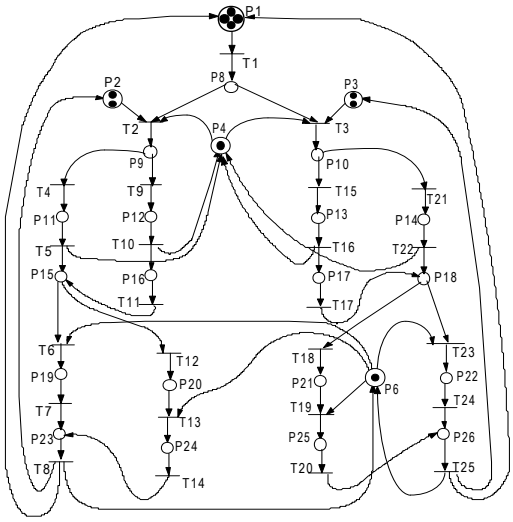



**Figura 5.10**: Troca de Escolha Sincronizada da fig. 5.9 por Exclusão Mútua

sincronizada (*choice-synchronization*), é possível contornar este problema, mas para isso será necessário modificar a estrutura da rede de Petri. A escolha–sincronizada pode ser definida como uma estrutura de exclusão–mútua (definição 2.21 na seção 2.3.2) na qual as escolhas foram previamente definidas e sincronizadas.

Para contornar este problema troca-se a estrutura de escolha–sincronizada pela estrutura de exclusão mútua, acrescentando-se à modelagem em MILP as restrições de precedência contidas na estrutura escolha–sincronizada.

Na figura 5.9 são mostradas duas RdP que contem este tipo de estrutura. Na figura 5.9a os lugares $p_2$, $p_6$, $p_{10}$ e $p_{11}$ formam um estrutura de escolha–sincronizada, assim como $p_3$, $p_4$, $p_5$, $p_7$, $p_8$, $p_9$, $p_{12}$ e $p_{13}$ formam outra. Esta RdP é conservativa e consistente,

possuindo um *bom funcionamento*, não sendo necessário alterá-la

Na RdP da figura 5.9b os lugares $p_4$ e $p_5$ e os lugares $p_6$ e $p_7$ formam estruturas de escolha sincronizada que ocasionam um mau funcionamento da rede, pois ao aplicar o algoritmo simplificado a esta rede, a matriz F resultante contem elementos negativos associados a estes lugares. Uma sugestão para resolver este problema é trocar a estrutura de escolha sincronizada por uma estrutura de exclusão mútua e acrescentando-se na modelagem em MILP a restrição de que uma operação aconteça antes da outra, levando-se em consideração a escolha sincronizada (mas como o sistema é periódico talvez isto não seja necessário). A figura 5.10 mostra um exemplo desta troca, onde os lugares $p_4$ e $p_5$ da figura 5.9 são trocados pelo lugar $p_4$ da figura 5.10, e também os lugares $p_6$ e $p_7$ da figura 5.9 são trocados pelo lugar $p_6$ da figura 5.10. Feita esta troca e aplicando-se novamente o algoritmo simplificado, verifica-se que a matriz F resultante contem apenas valores positivos ou nulos.

Se não for possível contornar o problema, a rede de Petri será considerada não tratável e o Algoritmo Geral é finalizado.

**R9** – Unificação de Fichas em um Componente Conservativo (Caso 3)

Se um componente conservativo contiver mais de um lugar marcado, deve ser possível através de alguns disparos de transições da rede mudar a marcação inicial destes lugares de tal forma que a rede contenha um e um único lugar marcado em cada um de seus componentes conservativos. Mas se isto não for possível, a rede de Petri não será viva e será considerada não tratável, conseqüentemente o Algoritmo Geral é finalizado.

**R10** – Ausência de um Componente Conservativo (Caso 4)

Se dois lugares ligados pela mesma transição, sendo um predecessor do outro, não fizerem parte de um mesmo componente conservativo, não será possível modelar a precedência entre estes lugares, pois a existência desta precedência juntamente com a ausência de um componente conservativo que inclua estes dois lugares, indica que existe um processo que não pertence a nenhum componente conservativo. Uma possível causa para isto é que este processo não é cíclico e neste caso é necessário criar um C-lugar que ligue o lugar de início do processo (o lugar que precede a todos) ao lugar de término do processo (o lugar que sucede a todos). Porém, se inclusão deste C-lugar não for possível ou se a causa deste problema (caso 4) não for a falta de um C-lugar, a rede de Petri será considerada não tratável, conseqüentemente o Algoritmo Geral é finalizado.

**R11** – Verificação de Transições nos Componentes Repetitivos Estacionários (Caso 5)

Após a aplicação do algoritmo simplificado à busca de componentes conservativos, aplica-se o algoritmo simplificado na busca de componentes repetitivos estacionários. Na verificação dos elementos não nulos da matriz G três casos podem acontecer:

1. Existência de alguma transição da rede que não pertença a nenhum componente estacionário da rede. Neste caso a rede de Petri não será repetitiva, nem será viva e nem reversível e por isso será considerada não tratável, conseqüentemente o Algoritmo Geral é finalizado.

2. Existência de t-semifluxo com elementos negativos. Neste caso, verifica-se para cada um de seus elementos negativos se existe em algum outro t-semifluxo o mesmo elemento, porém com valor positivo.
   (a) Se todos os elementos negativos tiverem algum correspondente positivo, a linha de G correspondente ao t-semifluxo $\alpha$ poderá ser modificada de forma a ter apenas elementos não negativos fazendo-se combinações lineares das linhas da matriz G. Neste caso, a rede será consistente e os elementos não nulos em $\alpha$ formarão um componente repetitivo estacionário.
   (b) Caso contrário a rede de Petri não será repetitiva, nem será viva e nem reversível e será considerada não tratável, conseqüentemente o Algoritmo Geral é finalizado.

As regras para verificação da adequação da RPP ao ESP que foram apresentadas nesta seção podem ser esquematizadas e aplicadas através do seguinte algoritmo:

**Algoritmo de Verificação da Rede de Petri:**

- Aplicar o algoritmo simplificado na busca de componentes conservativos para verificar as regras R7, R8, R9 e R10. Ao término do algoritmo, analisar a matriz F resultante, considerando as seguintes possibilidades:
  - Se algum componente conservativo não contém lugares marcados, aplicar regra R7.
  - Se algum elemento de F resultante é negativo, aplicar a regra 8.
  - Se algum componente conservativo possui mais um lugar marcado, aplicar a regra R9.
  - Se existem dois lugares ligados pela mesma transição, sendo um predecessor do outro, que não fazem parte de pelo menos um mesmo componente conservativo, aplicar a regra R10.
- Aplicar o algoritmo simplificado na busca de componentes repetitivos estacionários para verificar a regra R11.

Se algum destes caso ocorrer, cria-se um arquivo que mostre ao projetista onde está o problema.

### 5.3.3 Classificação dos lugares e decomposição em processos

Conhecidos os componentes conservativos e os componentes repetitivos estacionários é possível fazer a classificação dos lugares e dos processos da seguinte forma:

1. Os lugares marcados que foram incluídos na etapa anterior e que representam a condição de periodicidade dos processos são os C-lugares, os lugares sem marcação inicial são os A-lugares e os lugares marcados restantes são os B-lugares.
2. A identificação e a decomposição dos processos serão feitas da seguinte forma:
   (a) Se um componente conservativo contiver um C-lugar então este componente será um C-circuito ou uma C-junção (ver tabelas 4.2.2 e 3.1 nas seções 4.2.2 e 3.5.2, respectivamente). Se for um C-circuito será um processo ordinário (ver definição 2.26) ou será um processo simples de um processo com sincronismo. Se for uma C-junção corresponderá a combinações de processos simples em processos com alternância.

(b) Se o componente conservativo contiver um B-lugar então este componente será um B-circuito ou uma B-junção (ver tabelas 4.2.2 e 3.1). Se for um B-circuito será um processo ordinário ou será um processo simples de um processo com sincronismo. Se for uma B-junção corresponderá a um processo com compartilhamento.

Para cada B-junção deve-se verificar se o componente conservativo relacionado a ela é composto de operações simples ou de operações sequenciais (definidas na seção 4.3.2.2), esta informação será usada na montagem da formulação na próxima seção.

É necessário salientar que pela características dos processos com sincronismo, existe um componente conservativo para cada um de seus processos simples (ver seção 3.5.1), por isso para descobrir se um processo simples pertence a um processo com sincronismo, é necessário verificar se algum dos seus lugares pertence a outro componente conservativo, se isto ocorrer, é porque estes processos formam um processo com sincronismo.

No caso de processos com alternância, existe um mesmo p-invariante para todos os seus processos simples (ver seção 3.5.1). Por isso é necessário distinguir-se os processos simples contidos neles. O seguinte procedimento faz esta distinção:

- Para cada componente conservativo associado a uma C-junção (isto é, para cada processo com alternância), somar a quantidade de transições de entrada (SE) e as de saída (SS) de seus lugares. Seja SL a quantidade de lugares do componente conservativo e seja DI a seguinte diferença:

$$DI = (SE + SS) - 2 * SL$$

Esta diferença indica a quantidade máxima de processos simples que compõem o processo com alternância. A figura 5.11 mostra um processo com alternância com 4 processos simples.

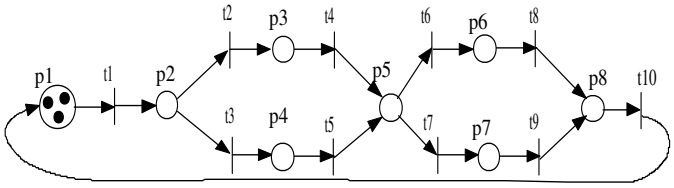



**Figura 5.11**: Processo com Alternância

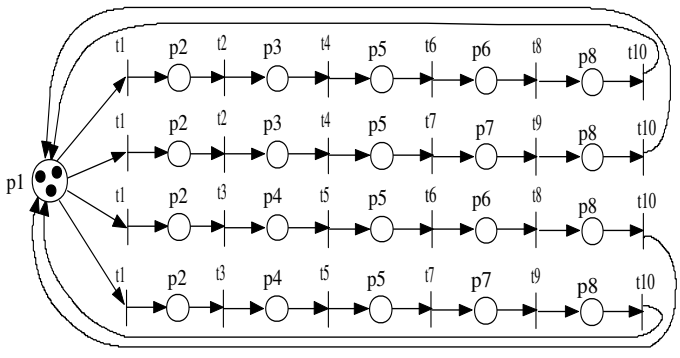



**Figura 5.12**: Processos Simples de um Processo com Alternância

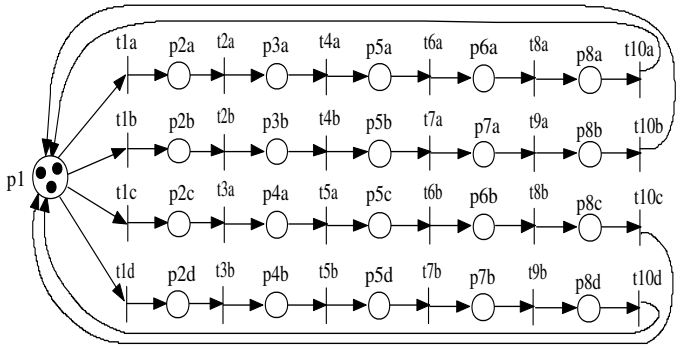



**Figura 5.13**: Lugares Renomeados

- Seja c o lugar marcado do processo com alternância. Separar os DI processos simples contidos neste processo. Isto pode ser feito através de um algoritmo de enumeração, explicitando todos os DI caminhos possíveis que ligam c a ele mesmo. Para ilustrar esta idéia a partir do exemplo da figura 5.11 chega-se a rede de Petri da figura 5.12.
- Para não haver confusão entre um caminho e outro é necessário renomear os lugares e as transições, mas sem perder o seu significado. Esta confusão ocorre na figura 5.12 com $p_2$, $p_5$ e $p_8$ sendo associados a 4 lugares distintos e $p_3$, $p_4$, $p_6$ e $p_7$ sendo associados a 2 lugares cada um. Para resolver esta confusão renomeia-se os lugares, como mostrado pela figura 5.13.

**Algoritmo de Classificação dos Lugares e de Decomposição dos Processos**

- Os lugares marcados usados para representar a condição de periodicidade dos processos são C-lugares, os lugares sem marcação inicial são A-lugares e os lugares marcados restantes são B-lugares.
- Verificar em cada componente conservativo, se o lugar marcado é um B- ou um C-lugar:
  - Se o componente conservativo contiver um C-lugar então este componente será um C-circuito ou uma C-junção:

    Se for um C-circuito verificar se algum de seus A-lugares pertencem a outro componente conservativo também. Se sim, estes componentes formam um processo com sincronismo; se não este componente forma um processo ordinário.

    Se for uma C-junção verificar quantos são os processos simples que fazem parte desta C-junção. Deve-se ainda verificar se algum de seus A-lugares também pertence a algum dos outros componentes conservativos. Se sim, estes componentes formam um processo síncrono de processos com alternância; se não este componente forma um processo com alternância.
  - Se o componente conservativo contiver um B-lugar então este componente será um B-circuito ou uma B-junção:

    Se for um B-circuito verificar se algum de seus A-lugares pertencem a outro componente conservativo também. Se sim, estes componentes formam um processo com sincronismo; se não este componente forma um processo ordinário.

Se for uma B-junção corresponderá a um processo com compartilhamento. Deve-se verificar se o componente conservativo associado a ela é composto apenas por operações simples ou se contém operações sequenciais. Deve-se verificar também se as operações foram renomeadas nos passos anteriores (verificação das C-junções). Se sim, estas modificações devem ser incluídas na B-junção.

### 5.3.4 Verificação das propriedades da rede de Petri

Feita a decomposição, é possível verificar as propriedades limitação, de reversibilidade e de vivacidade destas redes de Petri.

- Se a rede for coberta por um conjunto de componentes conservativos então ela será limitada.
- Se a rede puder ser decomposta em processos simples interligados apenas pelos B- e C-lugares, então ela será reversível e viva, pois sempre será possível retornar à marcação inicial.
- Se a rede tiver uma transição que não pertença a nenhum componente repetitivo estacionário então a rede não será reversível.
- Se a rede tiver um lugar que não pertença a nenhum componente conservativo então a rede não será viva.
- Se a rede não for consistente então ela não será reversível, de acordo com a propriedade 3.3 mostrada na seção 3.2.3.

Se a rede de Petri não for viva ou limitada ou reversível, ela será considerada não–tratável e o Algoritmo Geral é finalizado.

Sendo conhecidos os componentes da rede, como os A-, B- e C-lugares e seu relacionamento através dos processos simples, com sincronismo, com alternância e com compartilhamento, é possível fazer a formulação de escalonamento deste sistema cíclico. A próxima seção tratará deste assunto.

## 5.4 Escalonamento Ótimo de Sistemas Periódicos

A seção 4.3 tratou da modelagem de sistemas periódicos, através de uma formulação em programação linear inteira mista denominada Escalonamento de Sistemas Periódicos (ESP). Esta modelagem é mostrada a seguir (com $\rho_{ij} \subset J$, $\rho_{fg} \subset Q$, $J, Q \subset \mathbb{J}$ e $\forall b \subset B$ ):

$$\max \quad \varphi \tag{5.8}$$

$$\mu_{s_i} - \mu_i \geqslant \varphi\, l_{i,s_i} - O_i - \upsilon(1 - C_J),\ \forall i \in \mathbb{O}_J,\ \forall J \subset \mathbb{J}; \tag{5.9}$$

$$\mu_{Q1} - \mu_{Jn} \geqslant \varphi\, l_{n,1} - O_{Jn} z - \upsilon(2 - C_J - C_Q),\ J, Q \subset \mathbb{J}; \tag{5.10}$$

$$\mu_j = \mu_k,\ j = s_i,\ k = s_i,\ i \in \mathbb{O}_J, \mathbb{O}_Q,\ j \in \mathbb{O}_J,\ k \in \mathbb{O}_Q, j \neq k; \tag{5.11}$$

$$\mu_i - \mu_j \geqslant \varphi\, d_j - K_{ij} - \upsilon(1 - C_J);\ \forall i, j \in \rho_{ij}\ \ \rho_{ij} \subset \mathbb{O}_B; \tag{5.12}$$

$$\mu_i - \mu_g \geqslant \varphi d_g - K_{ig} - \upsilon(4 - X_{ib} - X_{gb} - C_J - C_Q), \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B, \forall b \subset B; \tag{5.13}$$

$$1 \leqslant K_{ij} + K_{fg}, \forall i \in \rho_{ij}, \forall g \in \rho_{fg}, \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; \tag{5.14}$$

$$
\begin{aligned}
1 &\geqslant K_{ig} + K_{fj}, \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; && (5.15)\\
1 &= K_{ij} + K_{fg} - K_{ig} - K_{fj},\ \forall \rho_{ij}, \rho_{fg} \subset \mathbb{O}_B; && (5.16)\\
\sum_{\forall m \subset B} X_{im} &= 1,\ \forall i \in \mathbb{O}_B, B \subset \mathbb{M}; && (5.17)\\
1 &\leqslant \sum_{\forall i \in Q} O_i \leqslant H_Q; && (5.18)\\
\sum_{\forall j \subset J} C_j &\geqslant 1; && (5.19)\\
\sum_{\forall i \in \mathbb{O}_J} \mu_{Ji} &\leqslant \upsilon\, C_J, \quad \forall J \subset \mathbb{J}; && (5.20)\\
0 \leqslant \mu_i &< 1, \quad i \in \mathbb{O}; && (5.21)\\
K, X, C \in \{0,1\}; &\quad O \in \{0,1,2\}; && (5.22)\\
\varphi &> 0.
\end{aligned}
$$

A seção 5.3.3 tratou da classificação dos lugares e da decomposição da rede em processos. De posse destas informações é possível passar da modelagem em redes de Petri para a modelagem em programação linear inteira mista da seguinte forma:

- Os A-lugares são as operações.
- O conjunto $\mathbb{J}$ corresponde a todos os processos ordinários, os com sincronismo e os com alternância do sistema a eventos discretos periódicos. Estes processos são os B- e C-circuitos e as C-junções da rede de Petri. A quantidade de fichas contidas nos B- e C-lugares destes processos representa a altura de recorrência do job a que eles pertencem.
- O conjunto $\mathbb{M}$ corresponde aos recursos utilizados pelos processos com compartilhamento do sistema a eventos discretos periódicos. Estes recursos são os B-lugares das B-junções. A quantidade de fichas contidas nestes recursos correspondem a quantidade de máquinas idênticas representadas por estee B-lugares.
- $\mu_i$ e $\mu_j$ representam os tempos de início de execução das operações $i$ e $j$.
- $s_i$ representa o sucessor imediato da operação $i$.
- $d_i$ representa a duração do processamento da operação $i$.
- $l_{i,s_i}$ é definida da seguinte forma (de acordo com a seção 4.3.2.1):

$$
l_{i,s_i} = \begin{cases} 0 & \text{se } i \text{ for a única operação associada a um recurso,} \\ d_i & \text{caso contrário.} \end{cases}
$$

- $h_i$ representa a altura de recorrência de $i$.
- $\sigma_{ij}$ significa que $j$ é o sucessor imediato da operação $i$.

- $k_{ij}$ representa uma disjunção entre as operações $i$ e $j$.
- $\rho_{ij}$ representa uma operação sequencial com a primeira operação sendo $i$ e a última sendo $j$.
- De acordo com a seção 4.3, o conjunto das inequações 5.9, 5.10,5.11, 5.19 e 5.20 modela processos ordinários, os com sincronismo e os com alternância. O conjunto de inequações 5.12, 5.13, 5.14, 5.15, 5.16 e 5.17 modela os processos com compartilhamento.
- Nas inequações 5.9, 5.10 e 5.11, cada subconjunto $J$ ou $Q$ do conjunto $\mathbb{J}$ corresponde a um $job$, os elementos $i$ e $j$ representam operações e o conjunto $\mathbb{O}_J$ é o conjunto de todas as operações deste $job$.
- Na inequação 5.18, $H_Q$ representa a altura de recorrência do $job$ $Q$.
- Quando $\mathbb{O}_J$ faz parte de um processo ordinário ou de um com sincronismo, o valor de $c_J$ é sempre um, pois o processo deverá ser executado. Consequentemente as inequações 5.19 e 5.20 são redundantes na modelagem destes processos. O mesmo não ocorre no caso do processo com alternância, pois a exigência é a de que pelo menos um dos seus processos simples seja executado (conforme discutido na seção 4.3).
- Na inequação 5.12, cada subconjunto $B$ do conjunto $\mathbb{M}$ corresponde a um processo com compartilhamento, e o conjunto $\mathbb{O}_B$ é o conjunto de todas as operações deste processo. Na inequação 5.15, cada $b$ corresponde a uma máquina simples do conjunto de máquinas paralelas $B$, a variável $X_{ib}$ representa a utilização da máquina $b$ pela operação $i$.

### 5.4.1 Escalonamento via GAMS

Tendo sido feita a formulação do sistema cíclico, o escalonamento de sistema cíclico poderá ser feito através do Sistema Geral de Modelagem Algébrica, **GAMS**, que é uma linguagem de alto nível para a formulação de modelos de Pesquisa Operacional. O programa GAMS/OSL resolve problemas de programação linear inteira mista através da técnica de *Branch & Bound*.

Devido a natureza combinatória de problemas em programação linear inteira mista, a sua resolução pode demandar um tempo grande, assim ao se utilizar o programa GAMS/OSL, pode-se estipular uma quantidade máxima de 1.500.000 iterações para se tentar achar a solução ótima de cada problema.

Exemplos de implementação em GAMS são mostrados no apêndice C.

# 6

# Exemplos

Este capítulo objetiva aplicar em três exemplos tirados da literatura, o procedimento geral para a determinação de um regime de operação cíclico para um sistema descritível por uma RPP, conforme proposto na seção 5.1.

O primeiro exemplo diz respeito a um sistema de controle distribuído baseado em Controladores Lógicos Programáveis (CLPs), apresentado no trabalho de Bonhomme *et. al.* (1999).

O segundo exemplo aborda o controle de um Sistema de Manufatura Flexível apresentado por Wang (1998). Em seu trabalho, Wang (1998) apresenta um modelo em rede de Petri para este sistema, porém com lugares não temporizados, por esta razão neste capítulo, os valores de duração das operações serão gerados de forma aleatória.

O terceiro exemplo trata de um simulador de sistemas de manufatura flexível modelado por Zhou & DiCesare (1993). Neste trabalho, Zhou & DiCesare (1993) apresentam um modelo em rede de Petri para este sistema, porém com lugares não temporizados, por esta razão, os valores de duração das operações serão gerados de forma aleatória.

## 6.1 Supervisão de um controle distribuído em muitos Controladores Lógicos Programáveis

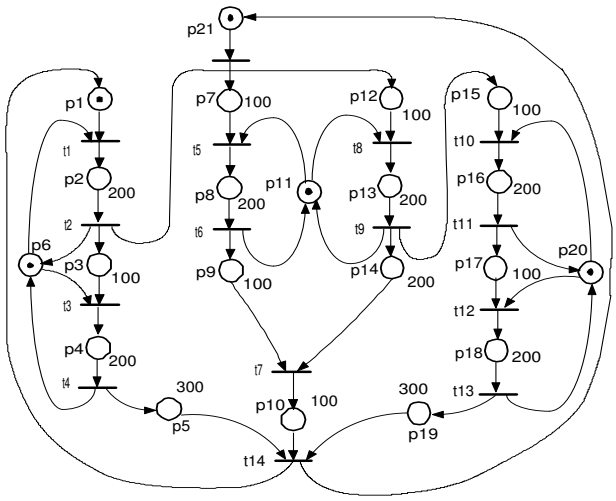



**Figura 6.1**: Exemplo de uma Supervisão de um controle distribuído em muitos CLPs

Esta seção tem o objetivo de apresentar a metodologia proposta no capítulo 5 através de um

exemplo simples, dado por um problema de supervisão de um controle distribuído em muitos controladores lógicos programáveis (CLPs). Este problema consiste em controlar as variáveis do sistema supervisionado (estas variáveis podem ser temperatura de um equipamento, pressão em uma caldeira, nível de um tanque, ...) mediante diferentes requisições aos CLPs. A duração das requisições depende de alguns fatores como o meio e o protocolo usados, a atividade do CLP, a quantidade de variáveis a serem lidas ou escritas. Geralmente, estas durações não são conhecidas com exatidão, mas são estimadas em um intervalo de tempo.

A figura 6.1 apresenta uma possível rede de Petri p-temporizada para modelar um problema de supervisão, consistindo de 6 diferentes requisições (simbolizadas por $(p_2, p_4, p_8, p_{13}, p_{16}$ e $p_{18})$) para três diferentes CLPs (simbolizados por $(p_6, p_{11}$ e $p_{20})$). Neste exemplo, a duração das operações (requisições) tem um valor fixo exato. Os tempos das operações são mostrados seguir.

| Operação | | Duração (u.t.) | Operação | | Duração (u.t.) |
|---|---|---|---|---|---|
| $p_2$ | = | 200 | $p_{12}$ | = | 100 |
| $p_3$ | = | 100 | $p_{13}$ | = | 200 |
| $p_4$ | = | 200 | $p_{14}$ | = | 200 |
| $p_5$ | = | 300 | $p_{15}$ | = | 100 |
| $p_7$ | = | 100 | $p_{16}$ | = | 200 |
| $p_8$ | = | 200 | $p_{17}$ | = | 100 |
| $p_9$ | = | 100 | $p_{18}$ | = | 200 |
| $p_{10}$ | = | 100 | $p_{19}$ | = | 300 |

Este modelo leva em conta:

- Restrições de precedência: por meio da seqüência $(p_2, t_2, p_3, t_3, p_4)$ a requisição associada a $p_4$ deve ser processada depois da que está associada a $p_2$.
- Restrições de sincronização: a transição $t_{14}$ assegura-se que a distância temporal entre a finalização de todas as requisições não exceda algum valor de tempo pré-definido.
- Restrições de compartilhamento de recursos: o CLP associado a $p_6$ não pode processar ao mesmo tempo as requisições $p_2$ e $p_4$, nem o CLP associado a $p_{11}$ pode processar ao mesmo tempo as requisições $p_8$ e $p_{13}$ e nem o CLP associado a $p_{20}$ não pode processar ao mesmo tempo as requisições $p_{16}$ e $p_{18}$.

A tabela de transições (tabela 6.1) mostra o significado de algumas transições no modelo inicial. Os outros significados podem ser facilmente deduzidos de forma semelhante.

### 6.1.1 Redução do modelo

Seguindo o algoritmo para redução de lugares mostrado na seção 5.2 tem-se:

- **Regra R1:** Não existem transições temporizadas, portanto a regra R1 não precisa ser aplicada.
- **Regra R2:** O único lugar marcado temporizado é o lugar $p_7$.

| | |
|---|---|
| $t_1$ | início do processamento da requisição 1 para o $CLP_1$ ($p_6$) |
| $t_2$ | término da requisição 1 e permissão para o processo |
| | da requisição 4 ($p_{13}$) no $CLP_2$ ($p_{11}$) e no $CLP_1$ ($p_6$) |
| $t_3$ | início do processamento da requisição 2 |
| $t_4$ | término da requisição 2 para o $CLP_2$ |

**Tabela 6.1**: Significado de algumas transições da figura 6.1

- Criar um lugar não temporizado $p_{21}$, com a mesma marcação de $p_7$ e com as mesmas transições de entrada.
- Eliminar os arcos de entrada do lugar $p_7$.
- Criar uma transição $t_{15}$ que seja ao mesmo tempo a única entrada do lugar $p_7$ e a única saída $p_{21}$.
- Retirar a marcação de $p_7$.

- **Regra R6:** Não existem ciclos próprios, portanto a regra R6 não precisa ser aplicada.
- **Regra R5:** Todos os lugares possuem transições de entrada, portanto a regra R5 não é aplicada.

**INÍCIO DO LAÇO**

*1ª iteração*

- **Regra R4:** Criar uma lista $L_1$ com todas as transições que possuam exatamente um lugar de entrada e um de saída.

$$L_1 = \oslash$$

como $L_1$ é uma lista vazia então R4 não será aplicada.

- **Regra R3:** Criar uma lista $L_2$ com todos os lugares que tenham exatamente as mesmas transições de entrada e de saída.

$$L_2 = \oslash$$

como $L_2$ é uma lista vazia então R3 não será aplicada (não existem lugares redundantes).

*2ª iteração*

- Como não foi necessário aplicar as regras R3 e R4, o algoritmo de redução e de transformação termina.

**FIM DO LAÇO**

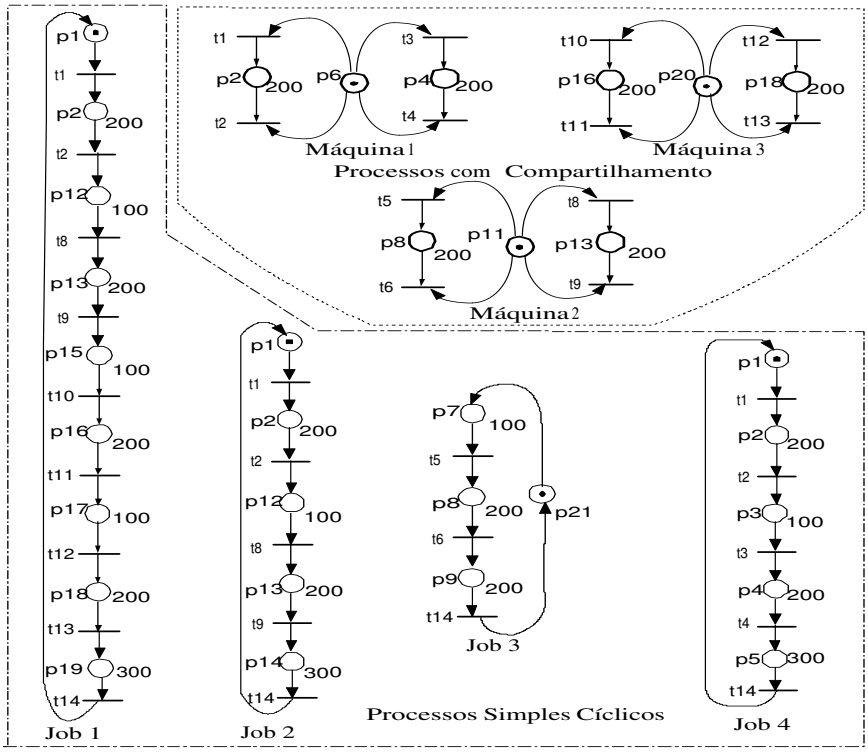



**Figura 6.2**: Invariantes da Rede de Petri reduzida

### 6.1.2 Busca dos Componentes

A partir da matriz de incidência C desta nova configuração e a matriz identidade F utiliza-se o algoritmo simplificado (seção 5.3.1), para encontrar os p-invariantes e os t-invariantes da rede de Petri reduzida. Os p-invariantes encontrados são mostrados pela figura 6.2.

### 6.1.3 Verificação das Propriedades de Sistemas Periódicos:

Seguindo o algoritmo para verificação da adequação da RPP ao ESP mostrado na seção 5.3.2 tem-se:

- **Regra R7:** A partir da matriz de incidência C da rede de Petri utilizando-se o algoritmo simplificado da seção 5.3.1 não são encontradas precedências conflitantes.
- **Regra R8:** A partir da matriz de incidência C da rede de Petri utilizando-se o algoritmo simplificado da seção 5.3.1 não são encontrados caminhos redundantes.
- **Regra R9:** Cada componente conservativo dado pelo algoritmo simplificada contém exatamente um lugar marcado.
- **Regra R10:** Não existem lugares interligados através da mesma transição que não estejam associados a pelo menos um mesmo componente conservativo.

- **Regra R11:** A partir da matriz de incidência $C'$ da rede de Petri utilizando-se o algoritmo simplificado da seção 5.3.1 para encontrar os componentes repetitivos estacionários da rede de Petri, verifica-se que cada transição da rede de Petri pertence a pelo menos um componente repetitivo estacionário.

Não existem mais transformações a serem feitas, passa-se agora para a classificação dos lugares usando-se o algoritmo mostrado na seção 5.3.1.

### 6.1.4 Classificação dos lugares

A figura 6.1 mostra a configuração do problema de CLP.
A classificação dos lugares é definida da seguinte forma:

- A-lugares: $p_2$, $p_3$, $p_4$, $p_5$ , $p_8$, $p_9$, $p_{10}$, $p_{12}$, $p_{13}$, $p_{14}$, $p_{15}$, $p_{16}$, $p_{17}$, $p_{18}$, $p_{19}$ e $p_{21}$;
- B-lugares: $p_6$, $p_{11}$ e $p_{20}$;
- C-lugares: $p_1$ e $p_{21}$.

### 6.1.5 Formulação do Problema de Escalonamento Cíclico

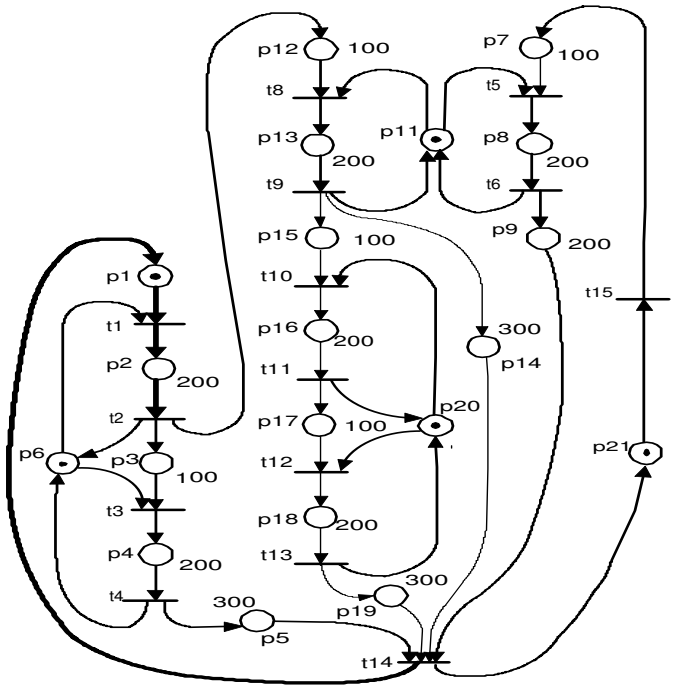



**Figura 6.3**: Rede de Petri p-temporizada do Sistema Cíclico

A redução da Rede de Petri e a classificação de lugares feitas nas seções anteriores resultou em um problema de *job shop cíclico* com 4 jobs distintos (processos simples cíclicos) e 3 máquinas (processos com compartilhamento), mostrados pela figura 6.2.

Estes sistema cíclico possui:

- 3 máquinas simples representadas pelos lugares $p_6$, $p_{11}$ e $p_{20}$;
- 4 jobs representados pelos C-circuitos job1, job2, job3 e job4;
- O job1 possui 8 operações, o job2 possui 4 operações, o job3 possui 3 operações e o job4 possui 4 operações. As relações de precedência e os tempos de cada operação são mostrados na figura 6.3.
- A altura de recorrência em cada *job* é igual a um, pois em cada C-lugar existe somente uma única ficha.

Para passar para a modelagem em programação linear inteira mista as seguintes associações devem ser feitas:

- Para cada uma das 19 operações representadas pelos lugares não marcados associa-se uma variável $\mu$, que representará o tempo de início desta operação e associa-se também uma variável O que relaciona esta operação a sua operação predecessora.
- Para cada um dos 4 *jobs* associa-se uma constante $H = 1$ que representará a altura de recorrência deste *job*.
- Para cada par de operações $(i, j)$ contido em uma máquina deve-se associar duas variáveis $K(i, j)$ e $K(j, i)$ que modelam a não–sobreposição destas operações na máquina.
- Como as máquinas são simples, as variáveis X não são necessárias.
- Como não existe processos com alternância, as variáveis C também não são necessárias.

Feitas as associações e usando-se a modelagem em programação inteira mista mostrada no capítulo 3 chega-se a seguinte formulação:

$$\min \varphi$$

*Job 1*

$$\mu_{15} - \mu_{13} \geqslant 200\varphi - O_{13}; \quad \mu_{16} - \mu_{15} \geqslant 100\varphi - O_{15}; \quad \mu_{17} - \mu_{16} \geqslant 200\varphi - O_{16};$$
$$\mu_{18} - \mu_{17} \geqslant 100\varphi - O_{17}; \quad \mu_{19} - \mu_{18} \geqslant 200\varphi - O_{18}; \quad \mu_2 - \mu_{19} \geqslant 300\varphi - O_{19};$$
$$O_2 + O_{12} + O_{13} + O_{15} + O_{16} \quad + O_{17} + O_{18} + O_{19} = 1;$$

*Job 2*

$$\mu_{12} - \mu_2 \geqslant 200\varphi - O_2; \quad \mu_{13} - \mu_{12} \geqslant 100\varphi - O_{12}; \quad \mu_{14} - \mu_{13} \geqslant 200\varphi - O_{13};$$
$$\mu_2 - \mu_{14} \geqslant 300\varphi - O_{14}; \quad O_2 + O_{12} + O_{13} + O_{14} = 1;$$

*Job 3*

$$\mu_9 - \mu_8 \geqslant 200\varphi - O_8; \quad \mu_{21} - \mu_9 \geqslant 200\varphi - O_9; \quad \mu_8 - \mu_{21} \geqslant 100\varphi - O_{21};$$
$$O_7 + O_8 + O_9 = 1;$$

*Job 4*

$$\mu_3 - \mu_2 \geqslant 200\varphi - O_2; \quad \mu_4 - \mu_3 \geqslant 100\varphi - O_3; \quad \mu_5 - \mu_4 \geqslant 200\varphi - O_4;$$
$$\mu_2 - \mu_5 \geqslant 300\varphi - O_5 \quad O_2 + O_3 + O_4 + O_5 = 1;$$

*Máquina 1*

$$\mu_4 - \mu_2 \geqslant 200 - K(4,2); \quad \mu_2 - \mu_4 \geqslant 200 - K(2,4)\varphi; \quad K(2,4) + K(4,2) \geqslant 1$$

*Máquina 2*

$$\mu_8 - \mu_{13} \geqslant 200 - K(13,8); \quad \mu_{13} - \mu_8 \geqslant 200 - K(8,13)\varphi; \quad K(8,13) + K(13,8) \geqslant 1$$

*Máquina 3*

$$\mu_{18} - \mu_{16} \geqslant 200 - K(16,18); \quad \mu_{16} - \mu_{18} \geqslant 200 - K(18,16)\varphi; \quad K(16,18) + K(18,16) \geqslant 1$$
$$\mu_i \geqslant 0; \ \varphi \geqslant 0; \qquad K \in \{0,1\}; \qquad O \in \{0,1,2\};$$

### 6.1.6 Escalonamento

A busca pelo escalonamento cíclico ótimo será feita através do programa GAMS, cujo algoritmo é mostrado na seção C.1 do apêndice C.

O ciclo ótimo encontrado foi de 1400 u.t. (unidades de tempo) e os tempos de início de cada tarefa são mostrados a seguir tendo como referência o primeiro período:

| Operação | | Momento de Início (u.t.) | Operação | | Momento de Início (u.t.) |
|---|---|---|---|---|---|
| $p_2$ | = | 0 | $p_{12}$ | = | 200 |
| $p_3$ | = | 200 | $p_{13}$ | = | 300 |
| $p_4$ | = | 300 | $p_{14}$ | = | 500 |
| $p_5$ | = | 500 | $p_{15}$ | = | 500 |
| $p_7$ | = | 1300 | $p_{16}$ | = | 600 |
| $p_8$ | = | 0 | $p_{17}$ | = | 800 |
| $p_9$ | = | 200 | $p_{18}$ | = | 900 |
| $p_{10}$ | = | 1200 | $p_{19}$ | = | 1100 |

O GAMS precisou de apenas 40 iterações para encontrar o escalonamento ótimo. Este resultado é apresentado na seção C.1.1 do apêndice C. Foram gerados:

- 13 blocos de equações que geraram 117 equações simples;
- 6 blocos de variáveis que geraram 51 variáveis;
- 289 elementos não nulos contendo 33 variáveis discretas.

## 6.2 Escalonamento Cíclico de um Sistema de Manufatura Flexível

O sistema de manufatura modelado por Wang (1998) é composto por três subsistemas: subsistema de processamento, de checagem e de reparo. Porém aqui será mostrado apenas o subsistema de processamento.

O subsistema de processamento é composto por 7 máquinas indicada por M1 a M7. Dois tipos de peças, indicadas como *W*1 e *W*2 são processadas e depois montadas como uma nova peça, indicada como *W*3. O fluxo do processamento é mostrado pela figura 6.4. *W*1 e *W*2 são processadas

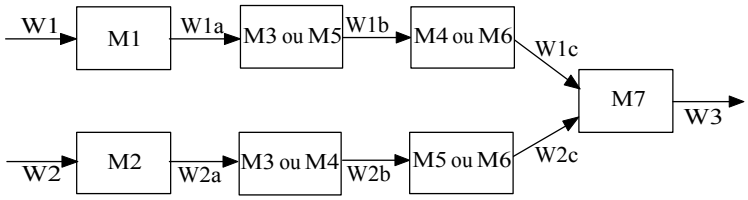



**Figura 6.4**: Fluxo de processamento dos subsistemas de processamento

em primeiro lugar pelas máquinas M1 e M2, resultando em *W*1a e *W*2a, respectivamente. A seguir, *W*1a é processada pela máquina M3 ou pela M5, resultando em *W*1b que deverá ser processada por M4 ou M6, que resultará em *W*1c; *W*2a é processada pela máquina M3 ou pela M4, resultando em *W*2b que deverá ser processada por M5 ou M6, que resultará em *W*2c. Finalmente, *W*1c e *W*2c são agregadas na máquina M7 resultando em *W*3 que é o produto do subsistema de processamento. Este subsistema pode ser modelado por redes de Petri. A figura 6.5 mostra a rede de Petri proposta para este subsistema por Wang (1998) e em seguida, o significado das transições e dos lugares desta rede de Petri é mostrado. Será mostrado também os tempos de duração dos lugares que são medidos em unidades de tempo (u.t.) e foram gerados aleatoriamente com valores no intervalo entre 10 e 70.

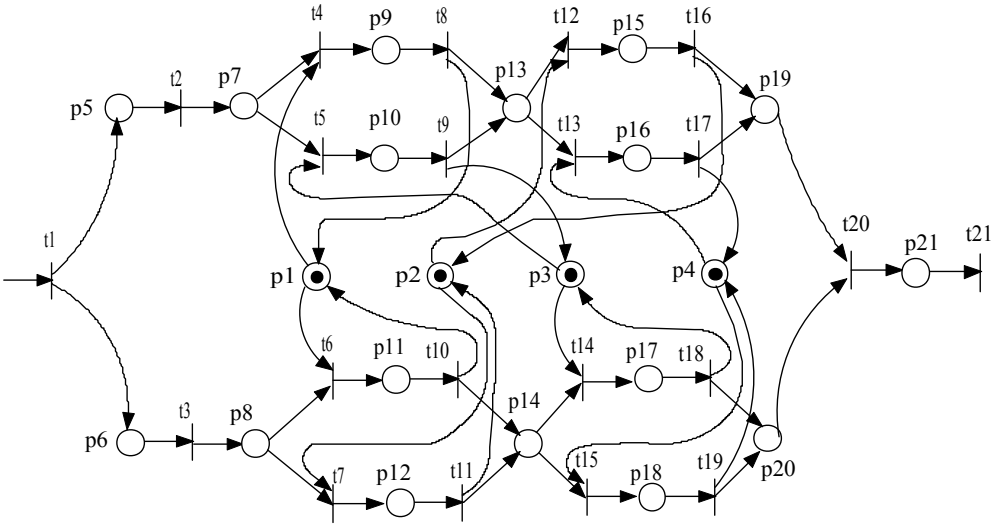



**Figura 6.5**: Rede de Petri para o Sistema de Manufatura Flexível

| Lugar | | Descrição | Tempo de Processamento |
|---|---|---|---|
| $p_1$ | : | M3 está disponível | 0 |
| $p_2$ | : | M4 está disponível | 0 |
| $p_3$ | : | M5 está disponível | 0 |
| $p_4$ | : | M6 está disponível | 0 |
| $p_5$ | : | W1 está sendo processada em M1 | 5 |
| $p_6$ | : | W2 está sendo processada em M2 | 5 |
| $p_7$ | : | W1a está pronto para ser processado por M3 ou M5 | 7 |
| $p_8$ | : | W2a está pronto para ser processado por M3 ou M4 | 8 |
| $p_9$ | : | W1a está sendo processada em M3 | 34 |
| $p_{10}$ | : | W1a está sendo processada em M5 | 43 |
| $p_{11}$ | : | W2a está sendo processada em M3 | 22 |
| $p_{12}$ | : | W2a está sendo processada em M4 | 33 |
| $p_{13}$ | : | W1b está pronto para ser processado por M4 ou M6 | 66 |
| $p_{14}$ | : | W2b está pronto para ser processado por M5 ou M6 | 12 |
| $p_{15}$ | : | W1b está sendo processada em M4 | 12 |
| $p_{16}$ | : | W1b está sendo processada em M6 | 10 |
| $p_{17}$ | : | W2b está sendo processada em M5 | 20 |
| $p_{18}$ | : | W2b está sendo processada em M6 | 22 |
| $p_{19}$ | : | W1c está pronto para ser processado por M7 | 12 |
| $p_{20}$ | : | W2c está pronto para ser processado por M7 | 21 |
| $p_{21}$ | : | W3 está sendo processada em M7 | 0 |
| $p_{22}$ | : | W1 e W2 estão disponíveis para serem processados | 0 |

| Transição | | Descrição |
|---|---|---|
| $t_1$ | : | Um par de peças são carregadas nas duas máquinas |
| $t_2$ | : | M1 finaliza o processamento W1 |
| $t_3$ | : | M2 finaliza o processamento W2 |
| $t_4$ | : | M3 começa a processar W1a |
| $t_5$ | : | M5 começa a processar W1a |
| $t_6$ | : | M3 começa a processar W2a |
| $t_7$ | : | M4 começa a processar W2a |
| $t_8$ | : | M3 finaliza o processamento W1a |
| $t_9$ | : | M5 finaliza o processamento W1a |
| $t_{10}$ | : | M3 finaliza o processamento W2a |
| $t_{11}$ | : | M4 finaliza o processamento W2a |
| $t_{12}$ | : | M4 começa a processar W1b |
| $t_{13}$ | : | M6 começa a processar W1b |
| $t_{14}$ | : | M5 começa a processar W2b |
| $t_{15}$ | : | M6 começa a processar W2b |
| $t_{16}$ | : | M4 finaliza o processamento W1b |
| $t_{17}$ | : | M6 finaliza o processamento W1b |
| $t_{18}$ | : | M5 finaliza o processamento W2b |
| $t_{19}$ | : | M6 finaliza o processamento W2b |
| $t_{20}$ | : | M7 começa a processar W1c e W2c |
| $t_{21}$ | : | M7 finaliza o processamento W3 |

Esta rede de Petri representa um sistema acíclico, para transformá-la em uma rede de Petri de sistema cíclico é necessário acrescentar um lugar que ligue as transições $t_1$ e $t_{21}$. Esta mudança é mostrada pela figura 6.6.

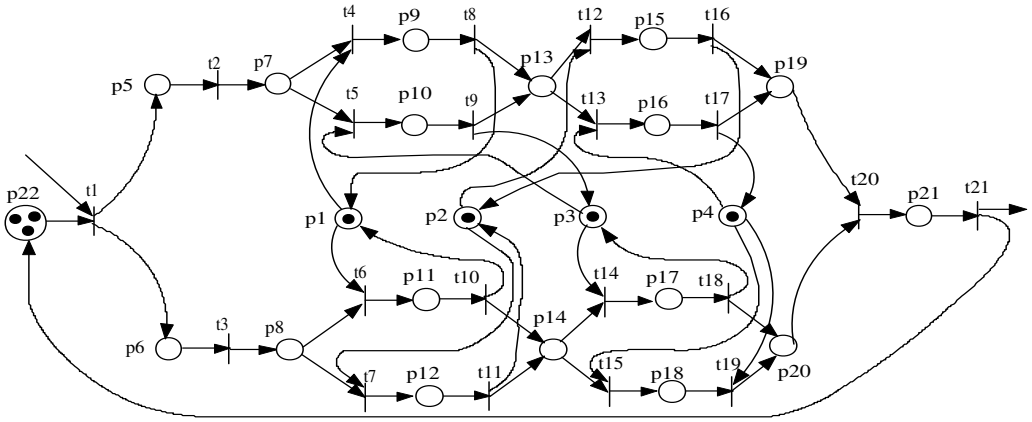



**Figura 6.6**: Rede de Petri para o Sistema de Manufatura Flexível Periódico

A diferença entre o momento de início do processamento das peças *W*1 e *W*2 e o momento de término do processamento de *W*3 representa o ciclo do sistema e a quantidade de fichas contidas no novo lugar $p_{22}$ estabelece a quantidade de peças *W*1 e *W*2 que podem ser processadas ao mesmo tempo pelo sistema. A quantidade de fichas em $p_{22}$ mostrada pela figura 6.6 é meramente ilustrativa, podendo ser qualquer valor inteiro maior que zero.

### 6.2.1 Redução do modelo

Não existem transições temporizadas (R1), nem lugares marcados temporizados (R2), nem lugares redundantes (R3), nem lugares de início vazios (R5) e nem ciclo próprio (R6). A aplicação do algoritmo de redução de lugares definido na seção 5.2, elimina os lugares $p_7$, $p_8$ e $p_{21}$ através de R4 (Fusão Serial), levando a rede de Petri reduzida mostrada pela figura 6.7, onde o lugar $p_{57}$ é resultado da fusão dos lugares $p_5$ e $p_7$, o lugar $p_{68}$ é resultado da fusão dos lugares $p_6$ e $p_8$, e os lugares $p_{192}$ e $p_{202}$ correspondem às fusões dos lugares $p_{19}$ e $p_{20}$ com o lugar $p_{21}$, respectivamente.

Pela análise de invariância feita através do algoritmo simplificado da seção 5.3.1, verifica-se que nesta rede de Petri não existem precedências conflitantes (R7) e nem caminhos redundantes (R8), e ainda que cada componente conservativo dado pelo algoritmo simplificada contém exatamente um lugar marcado (R9), não existem lugares interligados pela mesma transição que não estejam associados a pelo menos um mesmo componente conservativo (R10), e que cada transição da rede de Petri pertence a pelo menos um componente repetitivo estacionário (R11). Não existem mais transformações a serem feitas, passa-se agora para a classificação dos lugares usando-se o algoritmo mostrado na seção 5.3.1.

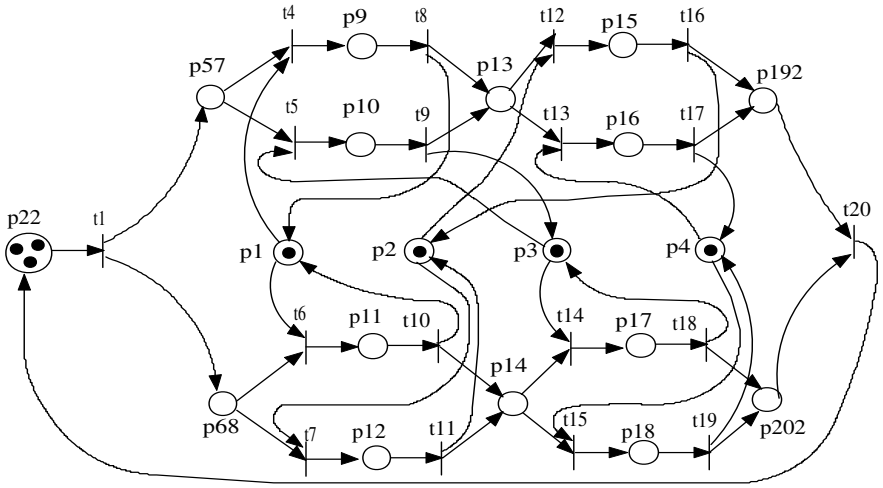



**Figura 6.7**: Rede de Petri Reduzida do Sistema

Os componentes conservativos da rede de Petri são mostrados pelas figuras 6.8 e 6.9. Note que o componente conservativo da figura 6.9 é composto por um processo com sincronismo que por sua vez é composto por 2 processos com alternância.

Antes de ser feita a formulação do problema é necessário decompôr os processos alternativos e renomear seus lugares, conforme discutido na seção 5.3.1. A figura 6.10 mostra todas as rotas possíveis para os processos com alternância, com os lugares auxiliares aux1, aux2, aux3 e aux4 sendo usados apenas para ilustrar a sincronização entre estes processos. Nesta decomposição os lugares já foram renomeados já sendo possível agora classificá-los usando-se o algoritmo mostrado na seção 5.3.1.

A aplicação deste algoritmo de classificação aos componentes conservativos encontrados anteriormente levam aos processos mostrados pela figura 6.10 e a seguinte classificação dos lugares:

- A-lugares: $p_5a$, $p_5b$, $p_5c$, $p_5d$, $p_{6a}$, $p_{6b}$, $p_{6c}$, $p_{6c}$, $p_{9a}$, $p_{9b}$, $p_{10a}$, $p_{10b}$, $p_{11a}$, $p_{11b}$, $p_{12a}$, $p_{12b}$, $p_{13a}$, $p_{13b}$, $p_{13c}$, $p_{13d}$, $p_{14}a$, $p_{14b}$, $p_{14c}$, $p_{14d}$, $p_{15a}$, $p_{15b}$, $p_{16a}$, $p_{16b}$, $p_{17a}$, $p_{17b}$, $p_{18a}$, $p_{18b}$, $p_{19a}$, $p_{19b}$, $p_{20a}$e $p_{20b}$ (ver figura 6.10);
- B-lugares: $p_1$, $p_2$, $p_3$ e $p_4$ (ver figura 6.8);
- C-lugar: $p_{22}$ (ver figura 6.6).

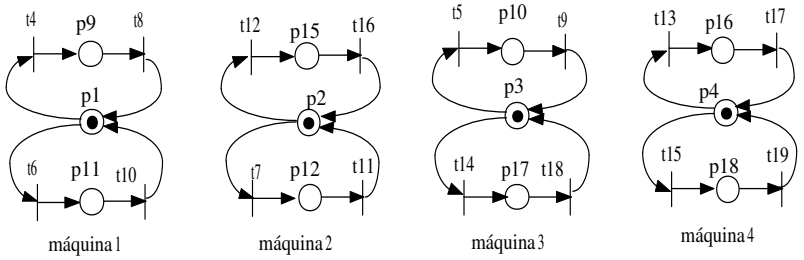



**Figura 6.8**: Processos com Compartilhamento

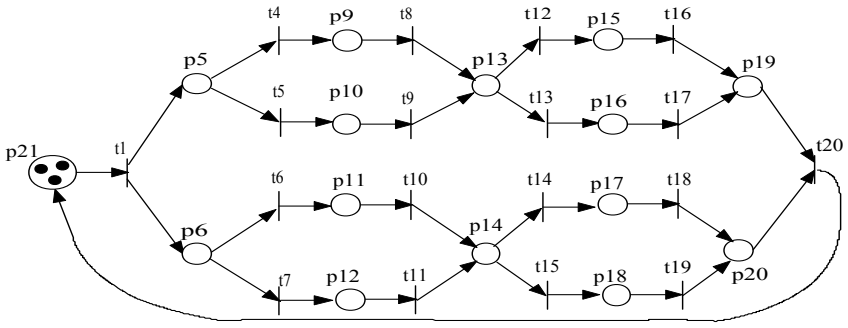



**Figura 6.9**: Processo Síncrono com 2 Processos com Alternância

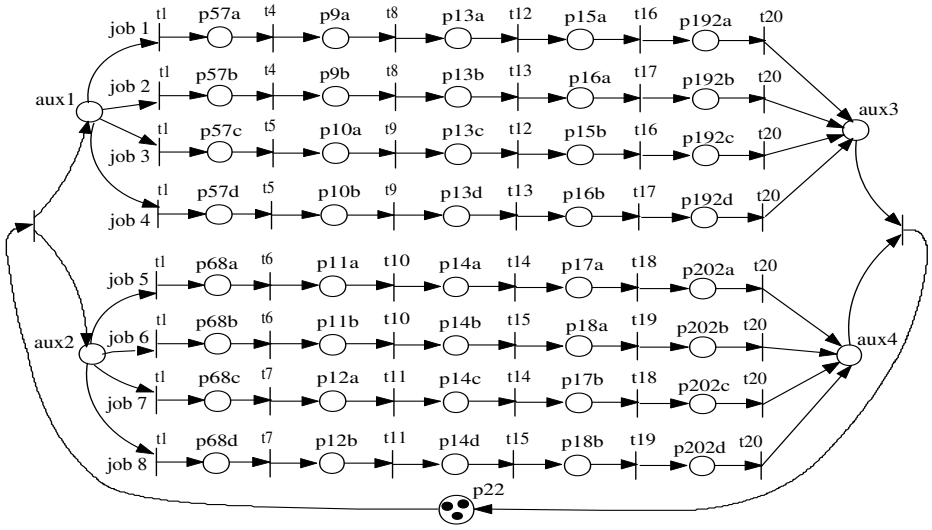



**Figura 6.10**: Processos Simples do Processos Composto por Sincronismo e Alternância

### 6.2.2 Formulação do Problema de Escalonamento Cíclico

A redução da Rede de Petri, a classificação de lugares e a decomposição dos processos feitas nas seções anteriores resultou em um problema de *job shop cíclico* com 8 jobs distintos mostrados pela figura 6.10 e 4 máquinas pela figura 6.8.

Este sistema cíclico possui:

- 4 máquinas simples representadas pelos lugares $p_1$, $p_2$, $p_3$ e $p_4$;
- 8 jobs representados por *job 1*, *job 2*, *job 3*, *job 4*, *job 5*, *job 6*, *job 7* e *job 8*;
- Cada *job* possui 5 operações, em um total de 40 operações. As relações de precedência destas operações são mostradas na figura 6.10 e os tempos de duração das operações são mostrados a seguir:

| Operação | | Duração | Operação | | Duração | Operação | | Duração |
|---|---|---|---|---|---|---|---|---|
| $p_5$ (a,b,c,d) | = | 12 | $p_{12}$ (a,b) | = | 33 | $p_{17}$ (a,b) | = | 20 |
| $p_6$ (a,b,c,d) | = | 13 | $p_{13}$ (a,b,c,d) | = | 66 | $p_{18}$ (a,b) | = | 22 |
| $p_9$ (a,b) | = | 34 | $p_{14}$ (a,b,c,d) | = | 12 | $p_{19}$ (a,b,c,d) | = | 12 |
| $p_{10}$ (a,b) | = | 43 | $p_{15}$ (a,b) | = | 12 | $p_{20}$ (a,b,c,d) | = | 21 |
| $p_{11}$ (a,b) | = | 22 | $p_{16}$ (a,b) | = | 10 | | | |

Conhecidos todos os componentes da formulação torna-se trivial, porém trabalhosa, a montagem das equações e inequações. Como a formulação será muito extensa não será colocada aqui neste trabalho.

### 6.2.3 Escalonamento

A busca pelo escalonamento cíclico ótimo foi feita através do programa GAMS mostrado na seção C.2 do apêndice C.

O ciclo ótimo encontrado foi de 134 u.t.. Apenas os *jobs* 3 e 5 são executados e os tempos de início das tarefas são mostrados pela tabela 6.2.3 tendo como referência o primeiro período. Como o sistema está inicialmente parado, são necessários dois ciclos, até o sistema entrar em regime.

| Operação | | Início | Operação | | Início | Operação | | Início |
|---|---|---|---|---|---|---|---|---|
| $p_5$ | = | 0 | $p_{12}$ | = | - | $p_{17}$ | = | 61 |
| $p_6$ | = | 0 | $p_{13}$ | = | 46 | $p_{18}$ | = | - |
| $p_9$ | = | 12 | $p_{14}$ | = | 49 | $p_{19}$ | = | 122 |
| $p_{10}$ | = | - | $p_{15}$ | = | - | $p_{20}$ | = | 81 |
| $p_{11}$ | = | 47 | $p_{16}$ | = | 112 | | | |

**Tabela 6.2**: Tempo de Início das Operações

O GAMS precisou de apenas 526 iterações para encontrar o escalonamento ótimo. Este resultado é apresentado na seção C.2.1 do apêndice C. Foram gerados:

- 17 blocos de equações que geraram 559 equações simples;
- 7 blocos de variáveis que geraram 154 variáveis;
- 2378 elementos não nulos contendo 112 variáveis discretas.

Note que como este exemplo lida com processos com alternância, a quantidade de blocos de equações e de blocos de variáveis é maior que no exemplo anterior. A quantidade de equações simples e de variáveis também aumentou com relação ao exemplo anterior, no entanto é interessante notar que as redes de Petri dos dois exemplos possui aproximadamente a mesma quantidade de lugares.

## 6.3 Aplicação em um Sistema de Manufatura Flexível

O Instituto Politécnico Rensselaer (*Rensselaer Polytechnic Institute*) desenvolveu um simulador físico, mostrado na figura 6.11, de um sistema de manufatura flexível (FMS)[1] em escala 1/6, a partir de um modelo de chão de fábrica automatizada que pode manufaturar uma família de produtos. Este simulador foi desenvolvido originalmente para ilustrar os benefícios de se construir um simulador físico em menor escala antes de se confiar em um modelo em escala real. Este simulador também é usado por pesquisadores para o projeto e implementação de controle para sistemas de manufatura automatizada usando redes de Petri.

### 6.3.1 Esquema de um Sistema de Manufatura Flexível

No sistema existem dois tipos de matéria bruta no estoque (bloco sólido e cavilha sólida). Estas matérias, em estado bruto, são manufaturadas e juntadas gerando um único produto. Uma parte da manufatura inicializa-se com um bloco sólido retangular que é fresado e perfurado em uma máquina fresadora com controle numérico computadorizado (CNC) e é finalizado como uma parte moldada geometricamente contendo um buraco cilíndrico em seu topo. A outra parte inicializa-se com uma cavilha sólida cilíndrica e é torneada em um torno CNC em uma cavilha de forma cônica/cilíndrica cuja base ajusta-se ao buraco na parte do bloco finalizado. A inserção robótica da cavilha no buraco do bloco é o processo de montagem. A família de produtos para este FMS compreende vários tamanhos e formatos de blocos e cavilhas montadas desta maneira. Os principais componentes do sistema ilustrado pela fig. 6.11 são:

1. A máquina fresadora e perfuratriz CNC.
2. O torno CNC.
3. O robô Microbot é um recurso compartilhado usado para carregar e descarregar materiais entre o torno CNC e a Esteira 3 (E3), e entre a máquina fresadora CNC e E2.
4. A Armazenagem Automatizada e o Sistema de Recuperação (AASR) é um armazém temporário (*buffer*) onde paletes[2] (*pallets*) com matérias em estado bruto e partes intermediárias podem ser armazenadas. A estrutura AASR é composta de locais de armazenagem $5 \times 4$ incluindo 19 compartimentos armazenador/palete utilizáveis. Um elevador transporta paletes carregados de e para o AASR.
5. O Sistema Automático de Transferência de Material (SATM) é um conjunto de recursos compartilhados que inclui quatro esteiras transportadoras de duas vias (um palete por vez) com sensores de presença em cada fim e um robô Guindaste (*Gantry*). O SATM é encarregado da transferência dos materiais entre as diferentes estações. Paletes universais são usadas, isto é, elas podem levar todos os materiais, suas partes e produtos. Robô Guindaste: transfere paletes, ora levando material do estoque ora levando uma parte finalizada, entre as quatro esteiras.

[1] flexible manufacturing system

[2] armação retângular de madeira para armazenagem de material, cuja parte inferior contem espaço suficiente para a inserção de garfos de empilhadeira.

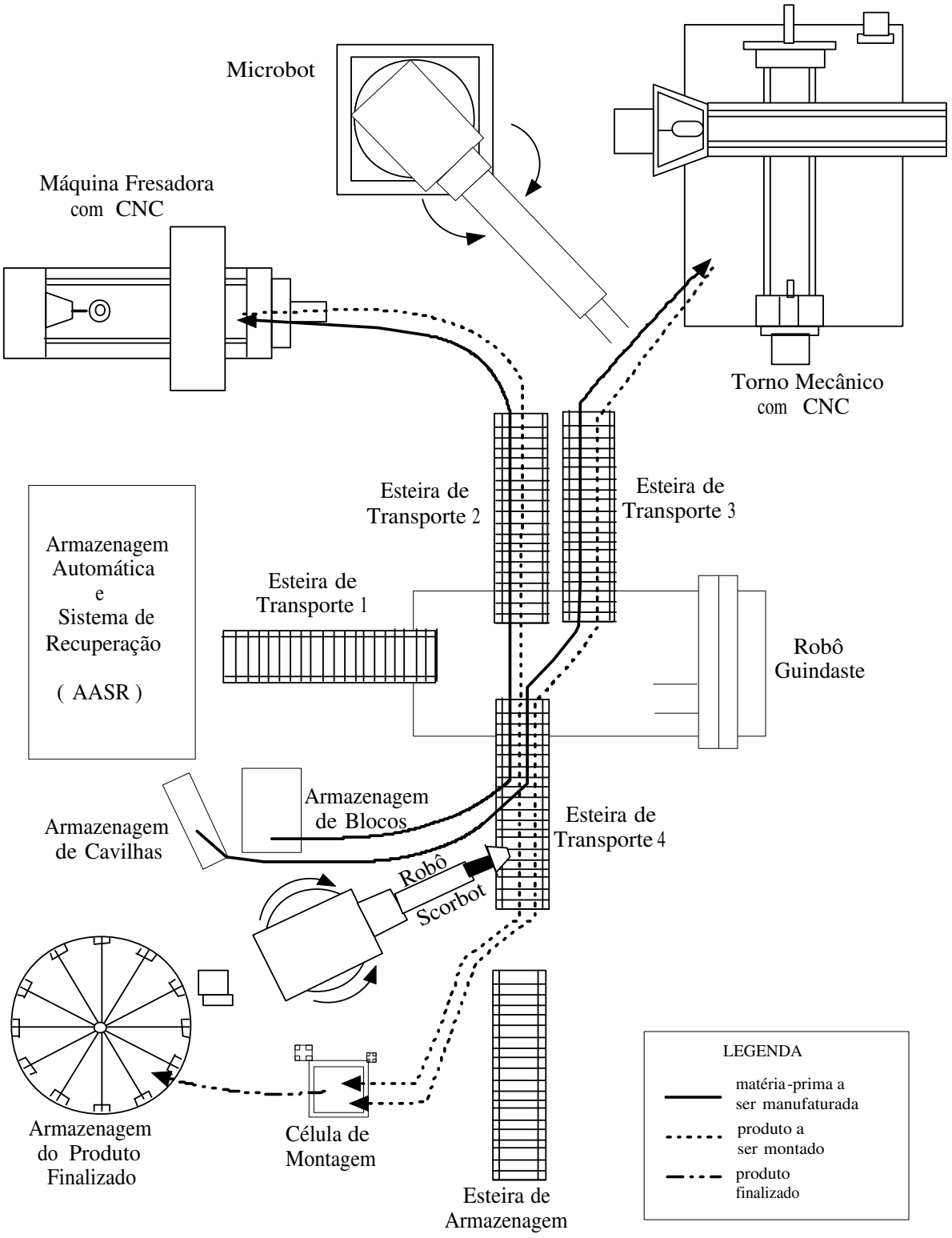



**Figura 6.11**: Sistema de Manufatura Flexível

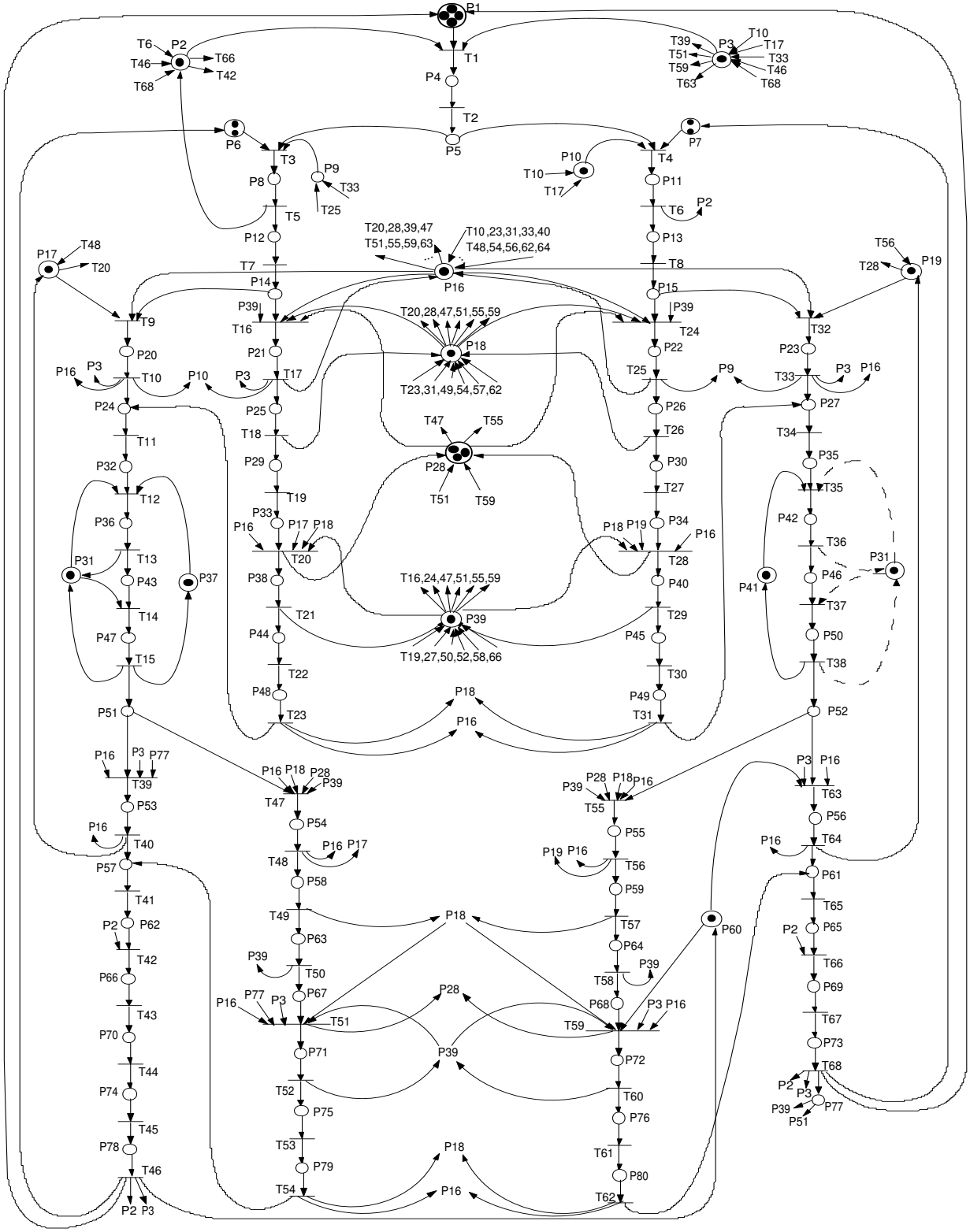



**Figura 6.12**: Rede de Petri do Sistema de Manufatura Flexível

E1 : Transfere paletes levando ou material em estoque
ou uma parte finalizada entre SATM e o robô Guindaste.
E2 : Transfere paletes levando ou material em estoque ou uma parte
finalizada entre a máquina fresadora CNC e o robô Guindaste.
E3 : Transfere paletes levando ou material em estoque ou uma parte
finalizada entre torno CNC e o robô Guindaste.
E4 : Transfere paletes levando ou material em estoque de sua área de
armazenagem para o robô Guindaste, ou uma parte finalizada
do robô Guindaste para a área de montagem.

6. O robô Scorbot é um recurso compartilhado com muitas funções: move paletes entre o depósito de paletes vazios e E4 em SATM; move cavilhas do depósito de cavilhas e blocos do depósito de blocos através em paletes sobre E4; move blocos e cavilhas em paletes sobre E4 para a célula de montagem e monta-os; e move o produto montado para o carrosel (depósito de produtos finalizados).

Zhou & DiCesare (1993) modelaram este sistema através da rede de Petri mostrada pela figura 6.12 cuja interpretação dos lugares é mostrada mais adiante. Existe um total de 68 transições que representa o início e o término das operações relacionadas. O tempo de duração associado a cada lugar foi escolhido aleatoriamente com um valor entre 1 e 100, sendo que aos lugares que representam disponibilidade de máquina ou a periodicidade de um processo associou-se tempo de duração nulo.

As seguintes abreviações serão usadas para apresentar a interpretação dos lugares:

| | | |
|---|---|---|
| cavilha em estado bruto (CEB) | - | palete carregado de CEB (palete CCEB) |
| bloco em estado bruto (BEB) | - | palete carregado de BEB (palete CBEB) |
| bloco finalizado(BF) | - | palete carregado de BF (palete CBF) |
| cavilha finalizada (CF) | - | palete carregado de CF (palete CCF) |

| Lugar | Descrição | Duração |
|---|---|---|
| $p_1$ | Esteira disponível | 0 |
| $p_2$ | Robô Scorbot disponível | 0 |
| $p_3$ | E4 disponível | 0 |
| $p_4$ | Mover um palete do armazém para E4 | 22 |
| $p_5$ | Esvaziar o palete em E4 | 30 |
| $p_6$ | CEB disponíveis | 0 |
| $p_7$ | BEB disponíveis | 0 |
| $p_8$ | Mover uma CEB do armazém para o palete em E4 | 21 |
| $p_9$ | Pegar um CEB | 0 |
| $p_{10}$ | Pegar um BEB | 0 |
| $p_{11}$ | Mover um BEB do armazém para o palete em E4 | 23 |
| $p_{12}$ | Baixar um palete CEB | 21 |
| $p_{13}$ | Baixar um palete BEB | 22 |
| $p_{14}$ | Um palete CEB em E4 | 12 |
| $p_{15}$ | Um palete BEB em E4 | 12 |
| $p_{16}$ | Robô guindaste disponível | 0 |
| $p_{17}$ | E3 disponível | 0 |
| $p_{18}$ | E1 disponível | 0 |
| $p_{19}$ | E2 disponível | 0 |
| $p_{20}$ | Mover um palete CCEB da E4 para a E3 | 35 |
| $p_{21}$ | Mover um palete CCEB da E4 para a E1 | 46 |
| $p_{22}$ | Mover um palete CBEB da E4 para a E1 | 26 |
| $p_{23}$ | Mover um palete CBEB de E4 para E2 | 58 |
| $p_{24}$ | Palete CCEB em E3 | 15 |
| $p_{25}$ | Mover um palete CCEB de E1 para ASRS | 47 |
| $p_{26}$ | Mover um palete CBEB de E1 para ASRS | 35 |
| $p_{27}$ | Palete CCEB em E2 | 47 |
| $p_{28}$ | ASRS tem espaço vazio | 0 |
| $p_{29}$ | Armazenagem do palete CCEB em ASRS | 58 |
| $p_{30}$ | Armazenagem do palete CBEB em ASRS | 76 |
| $p_{31}$ | Robô Microbot disponível | 0 |
| $p_{32}$ | Carregar um palete CCEB de E3 para o torno | 36 |
| $p_{33}$ | Paletes CCEB disponíveis | 0 |
| $p_{34}$ | Paletes CBEB disponíveis | 0 |
| $p_{35}$ | Carregar um palete CCEB de E2 para a fresa | 33 |
| $p_{36}$ | Carregar o torno com o CEB | 21 |
| $p_{37}$ | Torno disponível | 0 |
| $p_{38}$ | Recuperar um CEB de ASRS | 18 |
| $p_{39}$ | Elevador ASRS disponível | 0 |

| Lugar | Descrição | Duração |
|---|---|---|
| $p_{40}$ | Recuperar um CEB de ASRS | 98 |
| $p_{41}$ | Fresa disponível | 0 |
| $p_{42}$ | Carregar a fresa com um BEB | 45 |
| $p_{43}$ | Tornear um CEB | 23 |
| $p_{44}$ | Baixar um palete CCEB em E1 | 22 |
| $p_{45}$ | Baixar um palete CBEB em E1 | 86 |
| $p_{46}$ | Fresar um BEB | 89 |
| $p_{47}$ | Descarregar uma cavilha terminada em E3 | 43 |
| $p_{48}$ | Mover um palete CCEB de E1 para E3 | 78 |
| $p_{49}$ | Mover um palete CBEB de E1 para E2 | 32 |
| $p_{50}$ | Descarregar um bloco finalizado em E2 | 23 |
| $p_{51}$ | Baixar uma cavilha finalizada em E3 | 98 |
| $p_{52}$ | Baixar um bloco finalizado em E3 | 67 |
| $p_{53}$ | Mover um palete CCF de E3 para E4 | 45 |
| $p_{54}$ | Mover um palete CCEB De E3 para E1 | 65 |
| $p_{55}$ | Mover um palete CBEB De E2 para E1 | 43 |
| $p_{56}$ | Mover um palete CBF de E2 para E4 | 77 |
| $p_{57}$ | Palete CCF em E4 disponível | 0 |
| $p_{58}$ | Levar um palete CCF de E1 para ASRS | 83 |
| $p_{59}$ | Levar um palete CBF de E1 para ASRS | 32 |
| $p_{60}$ | Estação de montagem vazia | 0 |
| $p_{61}$ | Palete CBF em E4 disponível | 0 |
| $p_{62}$ | Retirar palete CCF de E4 | 32 |
| $p_{63}$ | Armazenar palete CCF em ASRS | 76 |
| $p_{64}$ | Armazenar palete CBF em ASRS | 54 |
| $p_{65}$ | Retirar palete CBF de E4 | 43 |
| $p_{66}$ | Instalar uma cavilha finalizada em um bloco na Célula de Montagem | 44 |
| $p_{67}$ | Paletes CCF em ASRS disponíveis | 0 |
| $p_{68}$ | Paletes CBF em ASRS disponíveis | 0 |
| $p_{69}$ | Colocar um bloco finalizado na Célula de Montagem | 11 |
| $p_{70}$ | Montar o produto cavilha-bloco | 88 |
| $p_{71}$ | Recuperar uma cavilha finalizada em ASRS | 96 |
| $p_{72}$ | Recuperar um bloco finalizado em ASRS | 10 |
| $p_{73}$ | Depositar um palete do tipo CBEB no armazém de paletes | 116 |
| $p_{74}$ | Elevador ASRS disponível | 0 |
| $p_{75}$ | Baixar palete CCEB em E1 | 17 |
| $p_{76}$ | Baixar palete CBEB em E1 | 14 |
| $p_{77}$ | Bloco finalizado na Célula de Montagem | 0 |
| $p_{78}$ | Levar um palete do tipo CCEB de E1 para E3 | 65 |
| $p_{79}$ | Mover palete CCF de E1 para E3 | 62 |
| $p_{80}$ | Mover palete CBF de E1 para E3 | 45 |

As fichas iniciais foram especificadas por Zhou & DiCesare (1993) da seguinte forma:

- Em $p_1$, $p_6$ e $p_7$ tem-se:
  $2 \leqslant m_0(p_1) \leqslant 20, \quad 1 \leqslant m_0(p_6) \quad \text{e} \quad 1 \leqslant m_0(p_7)$.
- Cada lugar $p_3$, $p_{10}$, $p_{16}$, $p_{17}$, $p_{18}$, $p_{19}$,$p_{31}$, $p_{37}$, $p_{39}$, $p_{41}$ e $p_{60}$ tem uma ficha.

- $m_0\,(p_{28}) = 19$.
- Os outros lugares inicialmente são não marcados.

### 6.3.2 Redução do modelo

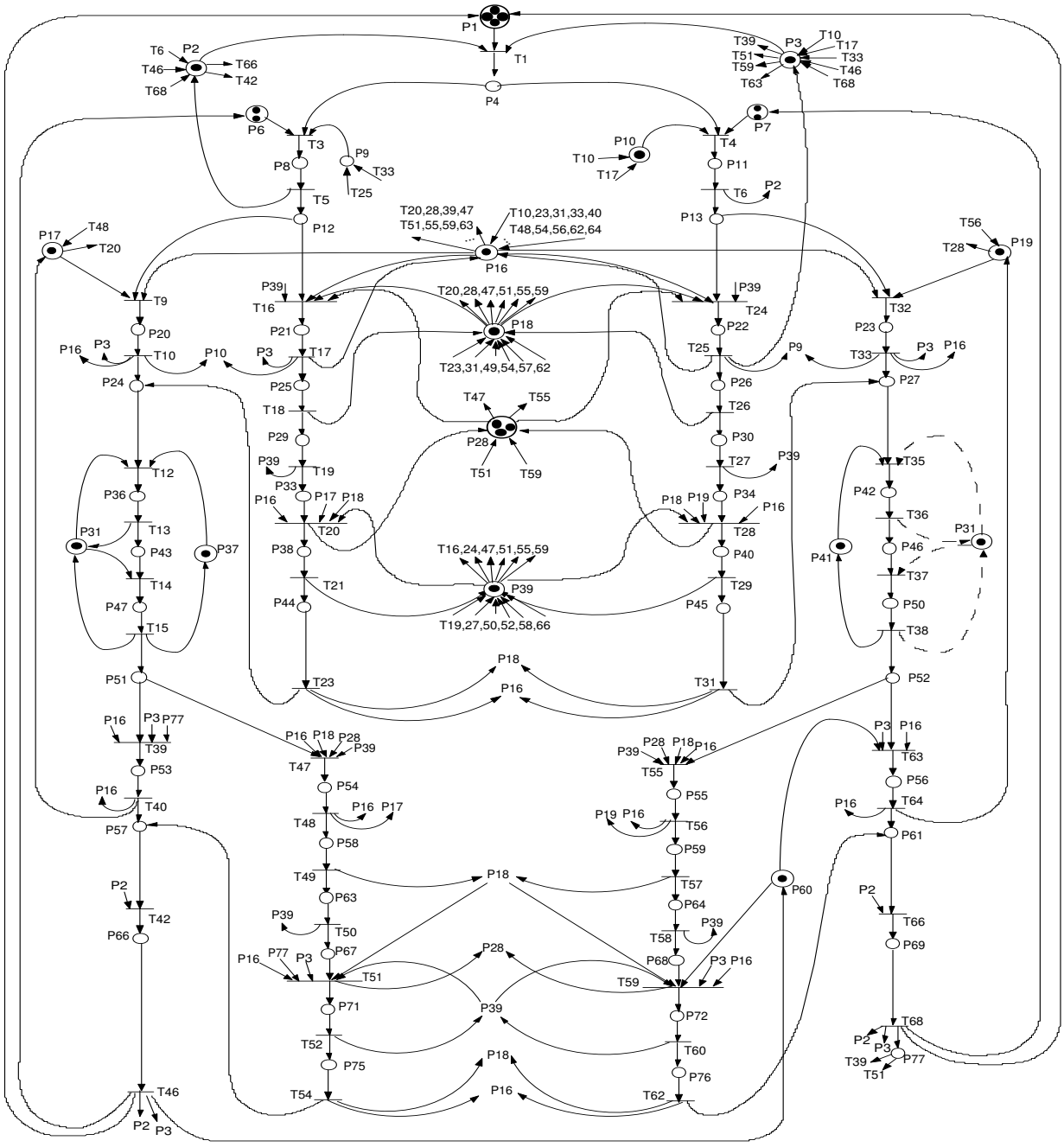



**Figura 6.13**: Rede de Petri Reduzida

Aplicando-se o algoritmo de redução encontra-se a rede de Petri mostrada pela figura 6.13.

Por esta figura nota-se que o algoritmo de redução eliminou os lugares $p_5$, $p_{14}$, $p_{15}$, $p_{32}$, $p_{35}$ , $p_{48}$, $p_{49}$, $p_{62}$, $p_{65}$, $p_{70}$, $p_{73}$, $p_{78}$, $p_{79}$ e $p_{80}$, através da fusão de lugares (R4).

## 6.4 Análise de Invariância

Ao aplicar o algoritmo para análise de invariância verificou-se pela regra R8.2, denominada de simplificação de caminhos redundantes, que esta rede é conservativa e é consistente apenas se for permitido que algumas operações aconteçam mais de uma vez no ciclo, o que não é permitido em nossa modelagem. Os caminhos redudantes acontecem devido a topologia dos lugares $p_9$ e $p_{10}$ e dos lugares $p_{60}$ e $p_{77}$ que representam uma estrutura de rede de Petri que é não abordada neste trabalho, denominada de escolha sincronizada (um exemplo desta estrutura foi mostrado pela figura 5.9). A escolha sincronizada impõe uma ordem de execução quando existe a possibilidade de escolha. Por exemplo, a transição $t_4$ que é a saída de $p_{10}$ deverá disparar sempre antes da transição $t_3$ que é a transição de saída de $p_9$. Isto significa que primeiro um bloco em estado bruto é processado, e só depois a cavilha em estado bruto poderá ser processada.

Para resolver este problema pode-se modificar a rede, transformando os lugares de escolha sincronizada em um único lugar de exclusão mútua. Isto é feito fundindo o lugar $p_9$ ao lugar $p_{10}$ e mantendo-se todas as suas transições de entrada e de saída. Da mesma forma, fundi-se os lugares $p_{60}$ e $p_{77}$. A figura 6.14 mostra a nova configuração da rede.

### 6.4.1 Classificação dos lugares

Os lugares são classificados da seguinte forma:

- A-lugares: $p_4$, $p_8$, $p_{11}$, $p_{12}$, $p_{13}$ , $p_{20}$, $p_{21}$, $p_{22}$, $p_{23}$, $p_{24}$, $p_{25}$, $p_{26}$, $p_{27}$, $p_{29}$, $p_{30}$, $p_{33}$, $p_{34}$, $p_{36}$, $p_{38}$ , $p_{40}$, $p_{43}$, $p_{44}$, $p_{45}$, $p_{46}$, $p_{47}$, $p_{50}$, $p_{51}$, $p_{52}$, $p_{53}$, $p_{54}$, $p_{55}$, $p_{56}$ , $p_{57}$, $p_{58}$, $p_{59}$, $p_{61}$, $p_{63}$, $p_{64}$, $p_{66}$, $p_{67}$, $p_{68}$, $p_{69}$, $p_{71}$, $p_{72}$, $p_{75}$ e $p_{76}$ (todos são lugares sem fichas mostrados na figura 6.14).
- B-lugares: $p_2$, $p_3$, $p_9$, $p_{16}$, $p_{17}$, $p_{18}$, $p_{19}$, $p_{28}$, $p_{31}$, $p_{39}$, $p_{37}$, $p_{41}$ e $p_{60}$ (todos são lugares com fichas em processos com compartilhamento, mostrados nas figuras 6.16 e 6.18 ).
- C-lugares: $p_1$, $p_6$ e $p_7$ (todos são lugares com fichas em processos simples ou com alternância, mostrados nas figuras 6.15, 6.16 e 6.17 ).
- B-circuito: mostrado pela figura 6.15.
- B-junções: mostradas pelas figuras 6.16 e 6.18.
- C-junções: mostradas pelas figuras 6.17.

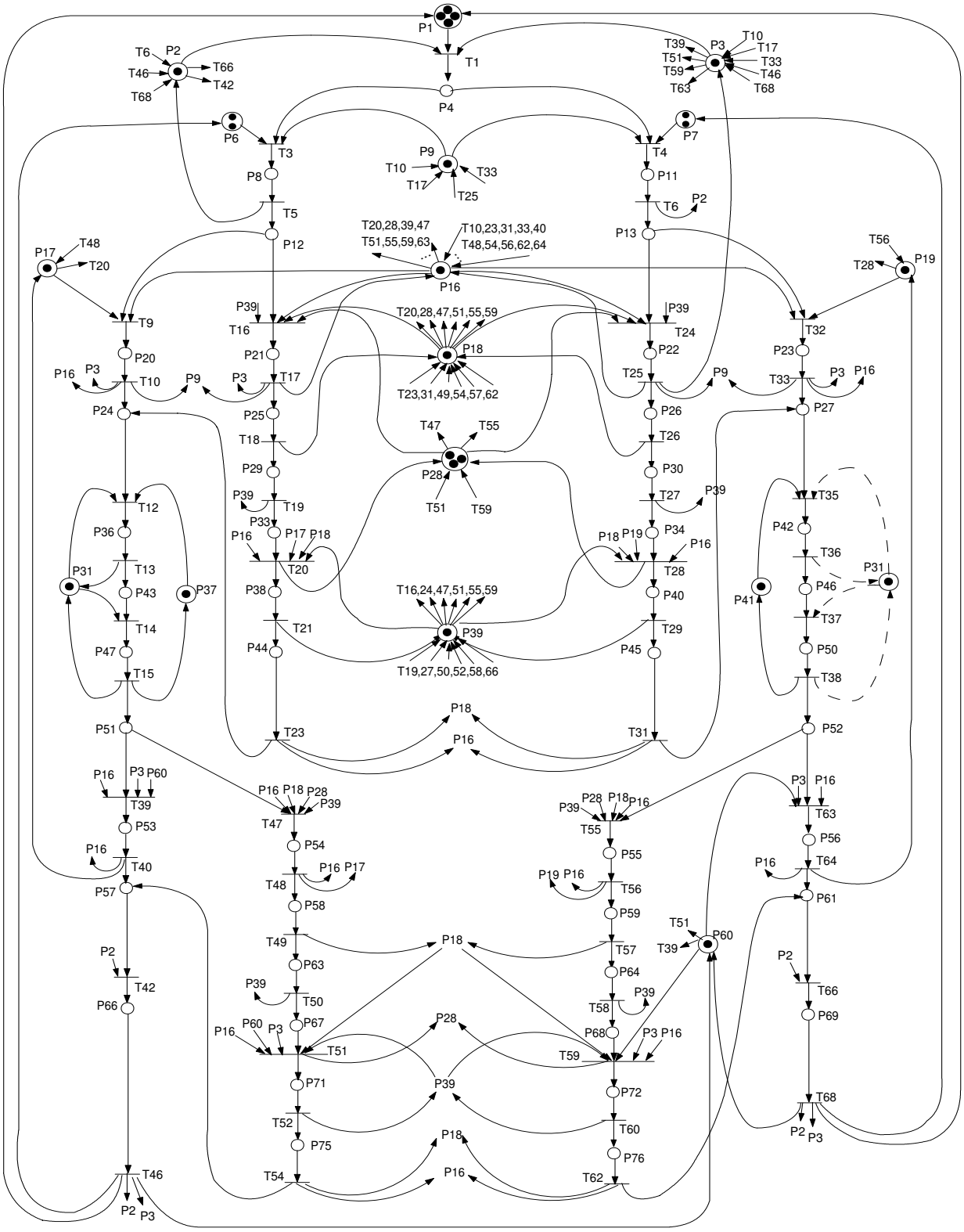



**Figura 6.14**: Nova Configuração da Rede de Petri

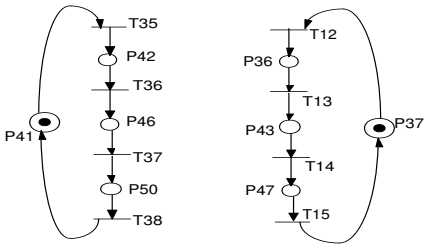



**Figura 6.15**: Processos Simples

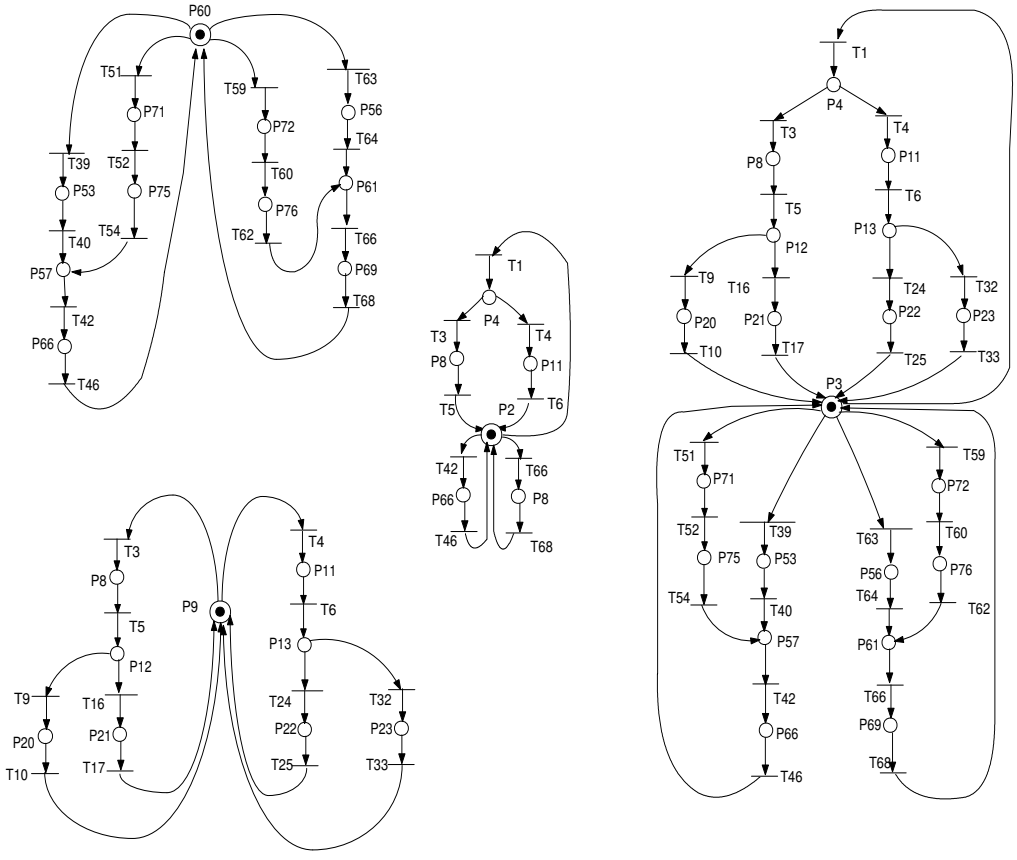



**Figura 6.16**: Processos Mistos – Compartilhamento e Alternância

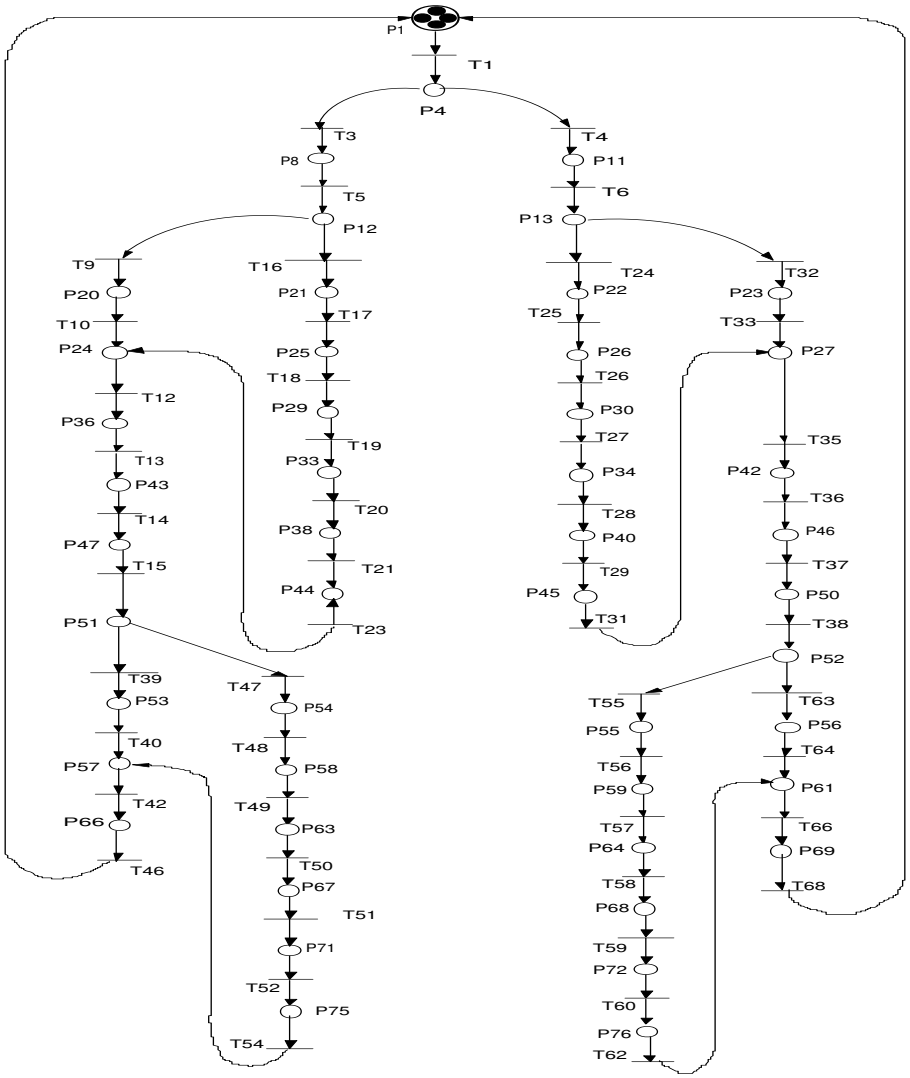



**Figura 6.17**: Processos com Alternância

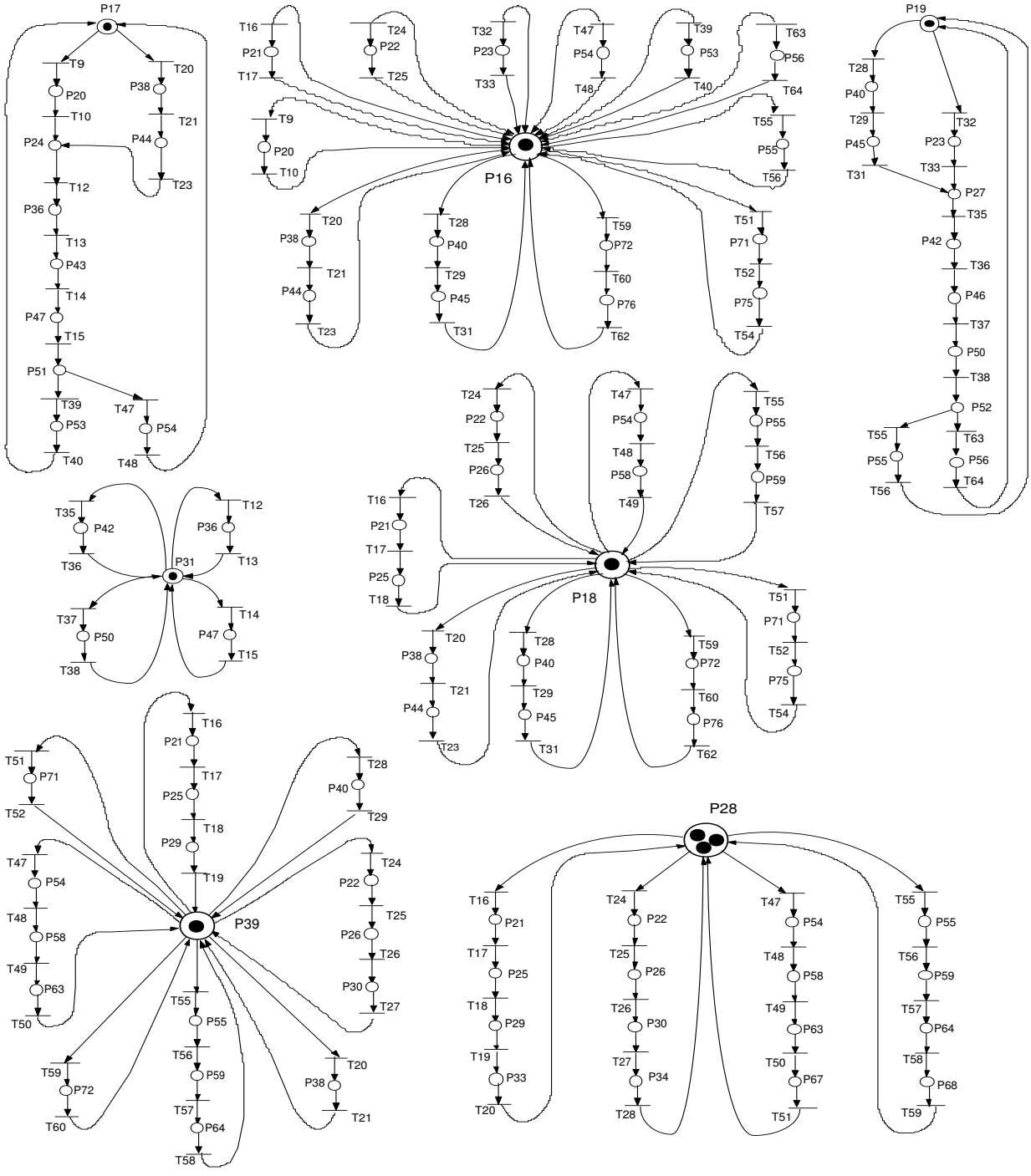



**Figura 6.18**: Processos com Compartilhamento

### 6.4.2 Formulação do Problema de Supervisão

A redução da Rede de Petri e a classificação de lugares feitas nas seções anteriores resultou em um problema com 24 jobs distintos (processos simples cíclicos) e 11 máquinas (processos com compartilhamentos), mostrados pelas figuras 6.15, 6.16, 6.17 e 6.18.

Estes sistema cíclico possui:

- 11 máquinas representadas pelos lugares $p_2$, $p_3$, $p_9$, $p_{16}$, $p_{17}$, $p_{18}$, $p_{19}$, $p_{28}$, $p_{31}$, $p_{39}$ e $p_{60}$. Estas máquinas podem ser simples ou paralelas, pois a quantidade de fichas nestes lugares pode ser alterada.

- 24 jobs gerados pelos processos simples (fig. 6.15) e pelos pelos processos com alternância (fig. 6.17);

- A altura de recorrência dos *jobs* de 1 a 8 será igual a 10, dos *jobs* de 9 a 16 será igual a 5 e dos *jobs* de 17 a 24 será igual a 1.

- As relações de precedência são mostradas na figura 6.14 e os tempos de cada operação são mostrados a seguir:

| Lugar | Inicio | Lugar | Inicio | Lugar | Inicio | Lugar | Inicio |
|---|---|---|---|---|---|---|---|
| $p_4$ | 52 | $p_8$ | 21 | $p_{11}$ | 23 | $p_{12}$ | 33 |
| $p_{13}$ | 34 | $p_{20}$ | 35 | $p_{21}$ | 46 | $p_{22}$ | 26 |
| $p_{23}$ | 58 | $p_{24}$ | 51 | $p_{25}$ | 47 | $p_{26}$ | 35 |
| $p_{27}$ | 80 | $p_{29}$ | 58 | $p_{30}$ | 76 | $p_{33}$ | 0 |
| $p_{34}$ | 0 | $p_{36}$ | 21 | $p_{38}$ | 18 | $p_{40}$ | 98 |
| $p_{42}$ | 45 | $p_{43}$ | 23 | $p_{44}$ | 100 | $p_{45}$ | 118 |
| $p_{46}$ | 89 | $p_{47}$ | 43 | $p_{50}$ | 23 | $p_{51}$ | 98 |
| $p_{52}$ | 67 | $p_{53}$ | 45 | $p_{54}$ | 65 | $p_{55}$ | 43 |
| $p_{56}$ | 77 | $p_{57}$ | 32 | $p_{58}$ | 83 | $p_{59}$ | 32 |
| $p_{61}$ | 43 | $p_{63}$ | 76 | $p_{64}$ | 54 | $p_{66}$ | 117 |
| $p_{67}$ | 0 | $p_{68}$ | 0 | $p_{69}$ | 27 | $p_{71}$ | 96 |
| $p_{72}$ | 10 | $p_{74}$ | 0 | $p_{75}$ | 79 | $p_{76}$ | 59 |
| $p_{77}$ | 0 | | | | | | |

### 6.4.3 Escalonamento

A busca pelo escalonamento cíclico ótimo foi feita através do programa GAMS mostrado na seção C.3 do apêndice.

Depois de 1921 iterações o GAMS conseguiu encontrar o escalonamento ótimo. O melhor ciclo factível encontrado foi de 157 u.t.. Apenas os *jobs* 1, 5, 9, 14, 17 e 22 são executados e os tempos de início de cada tarefa são mostrados a seguir tendo como referência o primeiro período, e como o sistema está inicialmente parado, são necessários sete ciclos, até o sistema entrar em regime:

| Lugar | Inicio | Lugar | Inicio | Lugar | Inicio | Lugar | Inicio |
|---|---|---|---|---|---|---|---|
| $p_4$ | 0 e 156 | $p_8$ | 135 | $p_{11}$ | 11 | $p_{12}$ | 0 |
| $p_{13}$ | 34 | $p_{20}$ | 44 | $p_{21}$ | - | $p_{22}$ | - |
| $p_{23}$ | 68 | $p_{24}$ | 79 | $p_{25}$ | - | $p_{26}$ | - |
| $p_{27}$ | 156 | $p_{29}$ | - | $p_{30}$ | - | $p_{33}$ | 115 |
| $p_{34}$ | - | $p_{36}$ | 156 | $p_{38}$ | - | $p_{40}$ | - |
| $p_{42}$ | 99 | $p_{43}$ | 20 | $p_{44}$ | - | $p_{45}$ | - |
| $p_{46}$ | 144 | $p_{47}$ | - | $p_{50}$ | 76 | $p_{51}$ | 86 |
| $p_{52}$ | 99 | $p_{53}$ | 156 | $p_{54}$ | - | $p_{55}$ | 9 |
| $p_{56}$ | - | $p_{57}$ | 124 | $p_{58}$ | - | $p_{59}$ | 52 |
| $p_{61}$ | 87 | $p_{63}$ | - | $p_{64}$ | 84 | $p_{66}$ | 156 |
| $p_{67}$ | - | $p_{68}$ | 156 | $p_{69}$ | 130 | $p_{71}$ | - |
| $p_{72}$ | 156 | $p_{74}$ | - | $p_{75}$ | - | $p_{76}$ | 9 |
| $p_{77}$ | - | | | | | | |

Este escalonamento leva em consideração que existem inicialmente 10 fichas no lugar $p_1$, 5 fichas nos lugares $p_6$ e $p_7$ e 19 fichas no lugar $p_{28}$, os outros lugares inicialmente marcados contem uma ficha cada (conforme descrição na seção 6.3.1).

Este resultado é apresentado na seção C.3.1 do apêndice C. Foram gerados:

- 20 blocos de equações que geraram 4999 equações simples;
- 7 blocos de variáveis que geraram 2786 variáveis;
- 17762 elementos não nulos contendo 2648 variáveis discretas.

Como este exemplo lida com processos com alternância, processos com sincronismo e operações sequenciais, a quantidade de blocos de equações e de blocos de variáveis é maior que dos exemplos anteriores. Ainda, a quantidade de equações simples e de variáveis é mais do que o quádruplo da dos exemplos anteriores, isto é, a quantidade aumentou não apenas porque a quantidade de lugares quadruplicou, mas também porque as relações entre estes lugares tornaram-se mais complexas.

Apesar do significativo aumento da complexidade, apenas 1921 iterações foram necessárias para que a solução fosse encontrada.

# 7

# Conclusões e Recomendações

## 7.1 Conclusões

Este trabalho teve como objetivo o estudo de uma classe (relativamente ampla) de sistemas a eventos discretos periódicos, com o desenvolvimento de ferramentas formais adequadas para a modelagem destes sistemas. Isto foi feito em dois níveis.

- No primeiro nível, foram estabelecidas relações funcionais entre os diversos componentes do sistema, foram apresentadas as redes de Petri (e algumas de suas extensões) como uma ferramenta para a representação, análise e controle de sistemas cíclicos e foram especificadas as propriedades essenciais a estas redes de Petri para que estas sejam definidas como redes de Petri de sistemas a eventos discretos periódicos.

- No segundo nível, buscou-se caracterizar a classe de sistemas a eventos discretos periódicos e, para esta classe, garantir um determinado procedimento de decisão, o escalonamento de operações e para a otimização do escalonamento cíclico do sistema foi proposta uma formulação em programação linear inteira mista.

Foram mostrados 3 exemplos, sendo um deles uma aplicação industrial, com o objetivo de ilustrar o método.

A metodologia proposta nesta tese para estudo de sistemas cíclicos apresenta uma série de vantagens e desvantagens que podem ser resumidas a seguir:

a) Vantagens

   i Modelar, analisar, planejar e controlar sistemas através de uma ferramenta de modelagem abrangente, com uma matemática bem definida e com fundamento prático.

   ii Possibilidade do mesmo modelo servir tanto para a análise quanto para o controle de sistemas.

   iii Através da resolução do escalonamento cíclico, é possível o estabelecimento de uma lei de controle, de modo a fazer o planejamento reativo para lidar com perturbações que possam ocorrer no sistema (Noronha & Santos–Mendes, 2002).

b) Desvantagens

i Pode ocorrer uma explosão combinatória de estados a medida que o sistema se torna mais complexo com aumento da quantidade de processos com compartilhamento ou com o aumento da quantidade de operações associadas a estes processos, conforme discutido na seção 4.3.3, nas considerações a respeito da complexidade do ESC.

ii A falta de um algoritmo eficiente e rápido para a solução do modelo em programação linear inteira mista do escalonamento cíclico de sistemas dificulta a utilização desta modelagem quando os sistemas são muito grandes.

## 7.2 Principais Contribuições

As principais contribuições deste trabalho são:

i Especificação das propriedades de Redes de Petri de Sistemas a Eventos Discretos Periódicos.

ii Elaboração de uma metodologia para a verificação destas propriedades.

iii Criação de uma modelagem em MILP para o problema de escalonamento cíclico destes sistemas.

iv Elaboração de uma modelagem mais abrangente do que a de Hanen (1994) e a de Rao (1992), incluindo os processos com sincronismo e os com alternância e as operações sequenciais.

## 7.3 Recomendações para futuros trabalhos

Este trabalho utilizou redes de Petri e uma modelagem em MILP, para modelar, analisar e controlar sistemas a eventos discretos periódicos, podendo-se observar que tal método foi bem apropriado para tratar deste tipo de sistema.

A metodologia pode se tornar mais completa, incluindo-se um controle para o comportamento transitório dos sistemas a eventos discretos periódicos, como estudado por Noronha & Santos–Mendes (2002) e por Leandro *et. al.* (2002).

A falta de um algoritmo eficiente e rápido para encontrar o melhor escalonamento cíclico de sistemas a eventos discretos periódicos, dificulta a utilização desta modelagem quando os sistemas são muito grandes e possuem muitas operações associadas a processos com compartilhamento, por isso é recomendável criar um algoritmo adaptado particularmente para este fim. Uma possibilidade seria a utilização de um algoritmo genético, dado que a classe de algoritmos genéticos possui um mecanismo de busca adaptativa que se baseia no princípio Darwiniano de reprodução e sobrevivência dos mais aptos, isto é, o método de busca mantêm a cada iteração uma população de soluções potenciais do problema que vão se adaptando ao longo das iterações, através de reprodução, cruzamento e mutação, o que equivale a percorrer várias áreas do espaço de busca a cada iteração, enquanto todos os outros métodos de busca processam apenas um único ponto no espaço de busca a cada iteração (Michalewicz, 1999). O algoritmo genético adaptado para a busca do escalonamento cíclico poderia explorar a estrutura combinatória do ESP e forneceria a cada iteração várias soluções factíveis do ESP (vários escalonamentos cíclicos), as melhores

soluções poderiam ser armazenadas ao longo das iterações, de forma que o projetista do sistema pudesse contar não apenas com a melhor solução obtida, mas também com outras soluções eventualmente satisfatórias.

# Referências Bibliográficas*


AALST, W.M.P. VAN DER. **Petri Net based Scheduling**. Computing Science Reports, Eindhoven University of Technology, 1995. **95**.

BAKER K.R. **Introduction to Sequencing and Scheduling**, John Wiley & Sons, 1974.

BALBO, G. *et. al.* **Introductory Tutorial – Petri Nets**. 21st International Conference on Application and Theory of Petri Nets, Aarhus, Denmark, 2000.

BERTHELOT, G. **Checking properties of nets using transformations**. Advances in Petri Nets (1985), Springer-Verlag, 1985, LNCS 222, 19–40.

BONHOMME, P.; AYGALINC, P.; CALVEZ, S. Transformation rules for P-Time Petri Nets. **Proceedings of 7th International Conference on Emerging Technologies and Factory Automation**, ETFA '99, 1999. Vol. 1, 251–260.

BOWDEN, F.D.J. **A Brief Survey and Synthesis of the Roles of Times in Petri Nets**. Mathematical and Computer Modelling, Elsevier, 2000. **31**, 55–68.

BRUCKER, P.; KAMPMEYER, T. Tabu Search Algorithms For Cyclic Machine Scheduling Problems. **Journal of Scheduling**, Springer-Verlag, 2005. **8**, 303–322.

CARDOSO, J.; VALETTE, R. **Redes de Petri**, Editora da UFSC, 1997.

CASSANDRAS, C. G.; LAFORTUNE S. **Introduction to Discrete Event Systems.** Kluwer. Boston, 1999.

CAVORY G.; DUPAS R.; GONCALVES G. A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints, **European Journal of Operational Research**, Vol. 161, Nº 1, 73–85, 2005


*Baseadas na norma NBR 6023, 2002, da Associação Brasileira de Normas Técnicas (ABNT)

CHAUVET, F.; HERRMANN, J.W.; PROTH, J.-M. Optimization of cyclic production systems: a heuristic approach. **In: IEEE Transactions on Robotics and Automation**, 2003. Vol. 19, Nº 1, 150–154.

COLOM, J.M.; SILVA M. Convex Geomery and Semiflows in P/T Nets: A Comparative Study of Algorithms for Computation of Minimal P-Semiflows. **Advances in Petri Nets** (1990), Springer-Verlag, 1991. LNCS 483, 79–112.

CRAMA, Y. *et. al.* Cyclic Scheduling in Robotics Flowshops. **Annals of Operations Research**, J.C Baltzer AG - Science Publishers, 2000. Vol. 96, 97–124.

CURY, J.E.R. Lançamento e Escalonamento de Tarefas em Sistemas de Manufatura com Alimentação Periódica. **Anais do 8º Congresso Brasileiro de Automação**, Belem–PA, Brasil, 1990. 1165–1171.

CURY, J.E.R. Teoria de Controle Supervisório de Sistemas a Eventos Discretos. **V Simpósio Brasileiro de Automação Inteligente**, Canela – RS, Brasil, 2001.

DAUSCHA, W.; MODROW H.D.; NEUMANN A. On Cyclic Sequence Types for Constructing Cyclic Schedules. **Zeitschrift für Operations Research**, 1985. **29**, 1–30.

DAVID, R.; ALLA H. Petri Nets for Modelling of Dynamic Systems. **Automática**, Prentice Hall, 1994. Vol. 30, Nº 2, 175–202.

DAWANDE, M. *et. al.*. Sequencing and Scheduling in Robotic Cells – Recents Developments. **Journal of Scheduling**, Springer Science, 2005. Vol. 8, 387–426.

DESROCHERS, A.A.; AL-JAAR R.Y. **Applications of Petri Nets in Manufacturing Systems**. IEEE Press. 1995.

DICESARE, F. *et. al.*. **Practice of Petri Net in Manufacturing**. Chapman & Hall, 1993.

FREEDMAN, P. Time, Petri nets, and robotics. **IEEE Transactions on Robotics and Automation**, 1991. Vol. 7, Nº 4, 417–433.

GEROGIANNIS, V.C.; KAMEAS A.D.; PINTELAS P.E. Comparative study and categorization of high-level Petri nets. **The Journal of Systems and Software**, 1998. Vol. 43, Nº 2, 133–160.

GULTEKIN, H.; AKTURK M. S.; KARASAN, O. Cyclic Scheduling of a 2-Machine Robotic Cell with Tooling Constraints, **European Journal of Operational Research**, 174 (2), 777-796, 2006.

HACK, M. **Analysis of Production Schemata by Petri Nets**. Master Thesis, Departament of Electrical Engineering, Masschussets Institute of Technology, Cambridge, Massachusetts,USA. 1972.

HANEN, C. Study of a NP-hard Cyclic Scheduling Problem: The Recurrent Job-Shop. **European Journal of Operational Research**, 1994. **72**, 82–101.

HILLIER, F.S.; LIEBERMAN G.J. **Introdução à Pesquisa Operacional**. Editora Campus, São Paulo, 1988.

HO, Y.-C. . Dynamics of discrete event systems. **Proceedings of the IEEE**, Preface to the special issue on discrete event systems, 1989, Vol. 77, 3–6.

HSU T.; DUPAS R.; GONÇALVES G.. A genetic algorithm to solving the problem of FMS cyclic scheduling. **IEEE International Conference on Systems, Man and Cybernetics**, 2002. Vol. 3.

ISO/IEC 15909-1:2004. **Software and system engineering - High-level Petri nets - Part 1: Concepts, definitions and graphical notation**. 2004.

ISO/IEC 15909-2:2004. **High-level Petri nets - Part 2: Transfer Format**, Working Draft, 2005.

JERBI, N. *et. al.*. Robust control of multi-product job-shops in repetitive functioning mode. **IEEE International Conference on Systems, Man and Cybernetics**, 2004. Vol. 5, 4917–4922.

KAMOUN, H.; SRISKANDARAJAH C. The Complexity of Scheduling Jobs in Repetitive Manufacturing Systems. **Eur. J. Oper. Res.**, 1993. Vol. 70, 35–364.

KATZ, V.; LEVNER, E. A Strongly Polynomial Algorithm for No-Wait Cyclic Robotic Flowshop Scheduling . **Operations Research Letters**, 1997. Vol. 21, 171–179.

KHANSA, W. **Réseaux de Petri P-temporel** - Contribution à l'etude des Systèmes à Evénements Discrets. PhD Thesis, Universitè de Savoie, 1997.

LAUTENBACH, K. Linear algebraic techniques for place/transition nets. **Lecture notes in Computer Science**, 1986. Vol. 254, 142–167.

LAWLER, E.L. **Combinatorial Optimization: Networks and Matroids**. Dover Publications, 1976.

LEANDRO, G. V. *et. al.*. Controller Synthesis for Cyclic Job Shop. In: 5TH Portuguese Conference on Automatic Control, 2002, Aveio. **Anais do 5TH Portuguese Conference on Automatic Control**, 2002.

LEE, T-E; POSNER M.E. Performance Measures and Schedules in Periodic Job Shops. **Oper. Res.**, 1997. Vol. 45, Nº 1, 72–91.

LEE, J.-K. A Scheduling Analysis in FMS Using the Transitive Matrix. In Artificial Intelligence and Simulation: 13th International Conference on AI, Simulation, Planning in High Autonomy Systems,**Lecture Notes in Computer Science**, 2005. Vol. 3397, 167–178.

LEVNER, E.; KATS V. A Parametric Critical Path Problem and an Aplication for Cyclic Scheduling. **Discrete Applied Mathematics**, 1998. Vol. 87, 149–158.

LUNA, H.P.L. Sistemas de Apoio à Decisão. **In: Manufatura Integrada por Computador**, Fundação CEFETMinas. 1995.

MEMMI, G.; ROUCAIROL G. Linear algebra in net theory. **Lecture notes in Computer Science**, 1980. Vol. 84, 213–224.

MOLLOY, M. K. Performance Analysis Using Stochastic Petri Net. **IEEE Transaction on Computer**, 1982. Vol. 11, Nº 4.

MOLLOY, M. K. Discrete time stochastic Petri Nets. **IEEE Transaction on Software Engineering**, 1985. Vol. 11, Nº 4.

MUNIER, A. The Complexity of a Cyclic Scheduling Problem with Identical Machines and Precedence Constraints. **European Journal of Operations Research**, 1996. Vol. 91, 471-480.

MURATA, T. Petri nets: Properties, analysis and applications. **Proceedings of the IEEE**, (1989). Vol. 77, Nº 4, 541–580.

NAKAMURA M. *et. al.*. Evolutionary Computing of Petri Net Structure for Cyclic Job Shop Scheduling. **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences** 2006. Vol. E89, Nº 11, 3235–3243.

NORONHA, A. B.; SANTOS–MENDES, R. Transitory Control em Cyclic Job Shop Scheduling. In: R. Boel; G. Stremersch. (Org.). Discrete Event Systems: Analysis and Control, **The Kluwer International Series in Engineering and Computer Science**. 1 ed. Bélgica: Kluwer Academic Publishers, 2000, v. 1, p. 371-382.

NORONHA, A. B.; SANTOS–MENDES, R. Controle de Transitórios em Escalonamento de Job Shop Cíclico.**Revista Controle & Automação**, 2002. Vol. 13, Nº 3, 231–247.

OOSTRUM, J. V. *et. al.* A Master Surgical Scheduling approach for cyclic scheduling in operating room departments. **21st European Conference on Operational Research in Iceland**, 2006

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

PETRI, C.A. **Kommunikation mit Automaten**. Ph.D. Dissertation, University of Bonn, West Germany. Tradução para o inglês: *Communication with Automata.* New York, Griffiss Air Force Base. Tech. Rep. RADC-TR-65-377, 1962. Vol. 1, Suppl.1, 1966.

PORTUGAL, D.S.V. **Estudo e Modelagem de Job Shop Cíclico com Jobs Distintos**. Dissertação de mestrado, Faculdade de Engenharia Elétrica e de Computação, Departamento de Controle e Automação Industrial, UNICAMP. 1999.

RAMADGE, P.J.G.; WONHAM W.M. The Control of Discrete Event Systems. **Proceedings of IEEE**, 1989. Vol. 77, Nº 1, 81–98.

RAO, U.S. **Multi-Stage, Identical Job Cyclic Scheduling for Repetitive Manufacturing**. Technical Report 1029, SORIE, Cornell University, Ithaca, NY, USA. 1992.

REISIG, W. **A Primer in Petri Net Design**. Springer, New York. 1992.

ROUNDY, R. Cyclic Schedules for Job Shops with Identical Jobs.**Mathematics of Operations Research**, 1992. Vol. 17, Nº 4, 842–865.

SANTOS–MENDES, R. Modelagem e Controle de Sistemas a Eventos Discretos. **In: Manufatura Integrada por Computador**, Fundação CEFETMinas. 1995.

SERAFINI, P.; UKOVICH W. A Mathematical Model for Periodic Scheduling Problems. **SIAM J. Disc. Math.**, 1989. Vol. 2. Nº 4, 550–581.

SIFAKIS, J. Performance Evaluation of Systems using Nets. Net Theory and Applications, Springer–Verlag, 1979. **LNCS**, Vol. 84 309–319.

SLOAN, H.R.; BUY, U. Reduction rules for Time Petri Nets. **Acta Informática**, Springer-Verlag, 1996. Vol. 33, Nº 7, 687–107.

SONG, J.S.; LEE T.E. A Tabu Search Procedure for Periodic Job Shop Scheduling. **Computers Ind. Engng**, 1996. Vol. 30, Nº 3, 433–447.

SONG, J.S.; LEE T.E. Petri Net Modeling and Scheduling for Cyclic Job Shops with Blocking. *Computers Ind. Engng*, 1998. Vol. 34, Nº 2, 281–295.

STEINER, G.; XUE, Z. On the connection between a cyclic job shop and a reentrant flow shop scheduling problem. **Journal of Scheduling**, Kluwer Academic Publishers, 2006. Vol. 9 , Nº 4, 381 – 387.

TROUILLET, B.; BENASSER, A. Cyclic scheduling problems with assemblies: an approach based to the search of an initial marking in a marked graph.**In: IEEE International Conference on Systems, Man and Cybernetics**, 2002. Vol. 3, página 6.

TSITSIKLIS, J.N. **On The Control of Discrete-Event Dynamical Systems**, MIT, Cambridge, Technical Report # LIDS-P-1661. 1987.

TUNCEL G.; BAYHAN G. M. A Survey of Scheduling in Manufacturing Systems with Petri Nets, **32nd International Conference on Computers and Industrial Engineering** – (ICC & IE), 2003.

VALETTE, R.; SILVA M. A Rede de Petri: Uma Ferramenta para a Automação Fabril. **Anais do 4º Congresso Nacional de Automação Industrial**, São Paulo – SP. 1990.

VERHAEGH, W.F.J. *et. al.* The Complexity of Multidimensional Periodic Scheduling. **Discrete Applied Mathematics**, 1998. **89**, 213–242.

WANG, J. **Timed Petri Nets - Theory and Application**. Kluwer Academic Publishers. 1998.

ZHOU, M.-C.; DICESARE F. **Petri Net Synthesis for Discrete Event Control of Manufacturing Systems**. Kluwer Academic Publishers. 1993.

ZURAWSKI, R.; ZHOU M. Petri Nets and Industrial Applications: A Tutorial.**IEEE Transactions on Industrial Electronics**, 1994. Vol. 41, Nº 6, 567–583.

# A

# Teoria de Grafos

Os grafos são objetos matemáticos clássicos cujas técnicas de representação são bem conhecidas. Um grafo é um diagrama composto de pontos (ou círculos), alguns dos quais são ligados entre si por linhas, e que é geralmente usado para representar graficamente conjuntos de elementos inter-relacionados. Os pontos, denominados nós, representam os elementos individuais; as linhas, chamadas arestas, representam a relação entre pares de elementos. Denominando-se por $N$ o conjunto de vértices da estrutura, isto é, os nós, e por $A$ o subconjunto dos pares ordenados do produto cartesiano $N \times N$ das ligações em $G$, istoé, as arestas, um grafo orientado é também representado por $G = (N, A)$. Um grafo é dito direcionado quando o sentido das ligações entre os vértices é importante. Nesse caso as arestas são chamadas de arcos. Ao invés de grafo direcionado, frequentemente usa-se a palavra digrafo, ou mesmo grafo se estiver claro no contexto que se trata de um digrafo. Como este capítulo só irá lidar com digrafos, eles serão tratados por grafos. Em geral, um grafo com fluxo de algum tipo em suas arestas é chamado de rede (Lawler, 1976).

Denote-se o número de nós por $n$, e o número do nó individual por$\{1, 2, \ldots, n\}$. Se $(i, j) \in A$, $i$ é chamado de nó inicial ou a origem do arco $(i, j)$, e $j$ o nó final ou o destino deste arco. Graficamente, os nós são representados por círculos, e o arco $(i, j)$ é representado por uma seta de $i$ para $j$.

A seguir será mostrada uma lista dos conceitos em teoria dos grafos, conforme Baccelli *et al.* (1992), mais relevantes para o nosso estudo.

- **Predecessor, sucessor:** Se em um grafo $(i, j) \in A$, então $i$ é dito predecessor de $j$ e $j$ é chamado de sucessor de $i$. O conjunto de todos os predecessores de $j$ é indicado por $\pi(j)$ e o conjunto de todos os sucessores de $i$ é indicado por $\sigma(i)$.

- **Origem, destino:** Se $\pi(i) = \emptyset$, o nó é chamado de origem; se $\sigma(i) = \emptyset$, o nó é dito ser um destino. Dependendo da aplicação, a origem é tratada por nó de entrada (respectivamente, nó de saída),ou pode ser chamada simplesmente de entrada (respectivamente, saída) de um grafo.

- **Caminho, circuito, laço, comprimento:** Um *caminho* $\rho$ é uma seqüência de nós $(i_1, i_2, \ldots, i_p)$, $p > 1$, tal que $i_j \in \pi(i_{j+1})$, $j = 1, \ldots, p-1$. O nó $i_1$ é o nó inicial e $i_p$ é o final deste caminho. Equivalentemente, pode-se dizer que um caminho é uma seqüência de arcos direcionados que conectam uma seqüência de nós. Um *caminho elementar* é um caminho em

que os nós aparecem apenas uma única vez. Quando o nó inicial e o nó final coincidem trata-se de um *circuito*. Um circuito $(i_1, i_2, \ldots, i_p = i_1)$ é um *circuito elementar* se o caminho $(i_1, i_2, \ldots, i_{p-1})$ é elementar. Um *laço* é um circuito $(i, i)$, isto é, um circuito composto por um único nó que é tanto inicial quanto final. Esta definição assume que $i \in \pi(i)$, isto é, que existe um arco de $i$ para $i$. O *comprimento* de um caminho ou um circuito é igual à soma dos comprimentos dos arcos dos quais ele é composto. Por convenção, o comprimento de um arco tem o valor um, mas pode assumir outro valor que deve ser especificado, nesse trabalho adotar-se-á sempre o valor convencional 1. Dessa forma, o comprimento do laço é igual a um. O comprimento do caminho $\rho$ é denotado $|\rho|_c$. A letra c subscrita refere-se a palavra comprimento. O conjunto de todos os caminhos e circuitos em um grafo é denotado por $R$. Um dígrafo é dito ser *acíclico* se $R$ não contém nenhum circuito.

- **Descendência, ascendência:** O conjunto de descendentes $\sigma^+(i)$ do nó $i$ consiste de todos os nós $j$ tal que exista um caminho de $i$ para $j$. Similarmente o conjunto de ascendente $\pi^+(i)$ do nó $i$ é o conjunto de todos os nós $j$ tal que exista um caminho de $j$ para $i$. A descendência pode ser descrita como $\pi^+(i) = \pi(i) \cup \pi(\pi(i)) \cup \ldots$. O mapeamento $i \mapsto \pi^*(i) = \{i\} \cup \pi^+(i)$ é o fechamento transitivo de $\pi$; o mapeamento $i \mapsto \sigma^*(i) = \{i\} \cup \sigma^+(i)$ é o fechamento transitivo de $\sigma$.

- **Subgrafo:** Dado um grafo $G = (N, A)$, um grafo $G' = (N', A')$ é dito ser um subgrafo de $G$ se $N' \subset N$ e se $A'$ consiste do conjunto de arcos de $G$ que que tem suas origens e seus destinos em $A'$.

- **Cadeia, grafo conectado:** Um grafo $G$ é dito conectado se para todos os pares de nós $i$ e $j$ em $G$ existe uma cadeia ligando $i$ a $j$. Uma cadeia é uma seqüência de nós $(i_1, i_2, \ldots, i_n)$ tal que entre cada par de nós sucessivos ou o arco $(i_j, i_{j+1})$ ou o arco $(i_{j+1}, i_j)$. Se as direções dos arcos forem desprezadas na definição de um caminho, obtem-se uma cadeia.

- **Grafo fortemente conectado:** Um grafo é dito ser fortemente conectado se para qualquer dois nós distintos $i$ e $j$ existe uma caminho de $i$ para $j$. Equivalentemente, $i \in \sigma^*(j)$ para todo $i, j \in G$, com $i \neq j$. Note que, de acordo com esta definição, um nó isolado, com ou sem um laço, é um grafo fortemente conectado.

- **Grafo bipartido:** Se o conjunto de nós $N$ de um grafo $G$ pode ser particionado em dois subconjuntos disjuntos $N_1$ e $N_2$, tal que cada arco de $G$ conecta um elemento de $N_1$ a um de $N_2$ ou vice-versa, então $G$ é um grafo bipartido.

# B

# O Job Shop Cíclico e sua Modelagem

Qualquer operação do conjunto de operações de um Job-Shop Cíclico (JSC) tem como principal característica a de ser processada infinitas vezes. Neste trabalho, assume-se que o conjunto de todas as operações, pode ser particionado exaustivamente em sub-conjuntos disjuntos e mutuamente exclusivos chamados de jobs (conforme Conway *et al.*, 1967). Cada job consiste, então, de um conjunto de operações que devem ser realizadas em alguma ordem. Assume-se que os atributos das operações incluem a associação a uma única máquina ($m_i$), tempo de processamento fixo ($p_i$), somente um único sucessor no job ($s_i$), uma vez que a operação começou a ser processada não pode ser interrompida para execução de outra operação, e ainda, as operações são não reentrantes*. A k-ésima inicialização e execução de todas as operações de um job será tratada aqui como a k-ésima iteração do job. O tempo transcorrido entre a inicialização da primeira operação de um job, em alguma iteração, e o término da execução da última operação do mesmo job e na mesma iteração será chamado de comprimento do job. O comprimento mínimo de um job será dado pela soma dos tempos de processamento de todas as operações deste job. Define-se também que o menor inteiro maior ou igual que a razão entre o comprimento do job e o comprimento do ciclo será tratado por largura do job e como o problema é periódico, a largura de um job é única, a partir do instante em que se determina um escalonamento cíclico. Assim tem-se:

$$\text{largura} = \left\lceil \frac{\text{comprimento do job}}{\text{comprimento do ciclo}} \right\rceil$$

Pode ser necessário controlar a largura dos jobs, a fim de se equilibrar o sistema, dado que é possível que existam jobs distintos, de comprimentos mínimos distintos, o que poderá acarretar em uma variação no comprimento e a largura destes jobs, tornando o sistema *injusto*, pois poderá haver muitos mais exemplares prontos de um *job* do que de outro. O exemplo a seguir ilustra bem esta idéia.

**Exemplo B.1** *São dados três* jobs, *A, B e C, com 5, 10 e 9 operações respectivamente, obedecendo uma rota pré determinada descrita pela tabela B.*

Tendo como objetivo minimizar o período para a repetição das operações, o período ótimo sem o controle de largura é de 76 unidades de tempo e o escalonamento ótimo desse caso é mostrado através do gráfico de Gantt da solução em regime apresentado na fig. B.1.

*As operações não se sobrepõem.

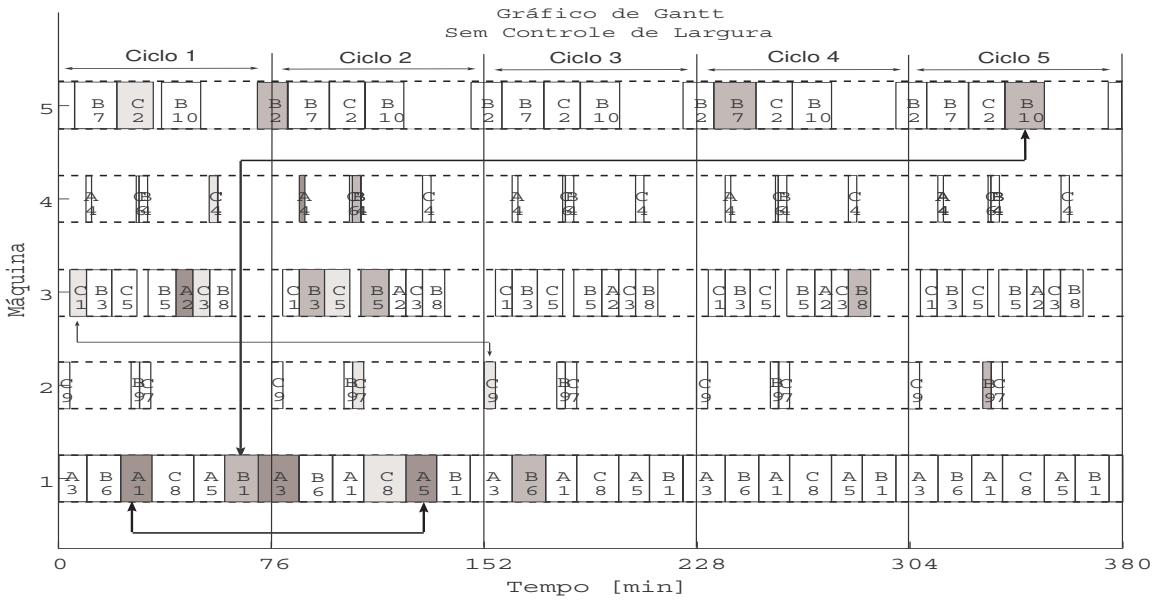



**Figura B.1**: Exemplo de um Job-Shop Cíclico Sem Controle de Largura

As setas indicam o comprimento dos jobs A e B, assim pode ser notado que qualquer iteração do job B permanece no sistema por um período bem maior que qualquer iteração do job A, o que poderia ser natural visto que B tem uma duração mínima de 97 u.t., enquanto A tem uma de 45 u.t.. Usando-se a restrição de controle de largura, para que cada instância de qualquer job tenha largura máxima igual a 2 ciclos, pode-se verificar na figura B.2 que a cada 2 ciclos alguma iteração dos jobs A, B e C é finalizada, sendo que o período ótimo continua a ser de 76 unidades de tempo.

Um atributo, que apresenta uma forte correspondência com a largura, é o da altura de recorrência entre operações, tanto em relação às instâncias de um único job (Hanen, 1994), quanto em relação aos jobs distintos. A altura de recorrência entre duas operações $i$ e $j$ é melhor descrita

| Job | Operação | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | (tempo de processamento da operação i, máquina associada à operação i) | | | | | | | | | |
| A | (11,1) | (6,3) | (15,1) | (2,4) | (11,1) | - | - | - | - | - |
| B | (12,1) | (11,5) | (9,3) | (3,4) | (10,3) | (12,1) | (15,5) | (8,3) | (3,2) | (14,5) |
| C | (6,3) | (13,5) | (6,3) | (3,4) | (9,3) | (1,4) | (4,2) | (15,1) | (4,2) | - |

**Tabela B.1**: Dados do exemplo B.1

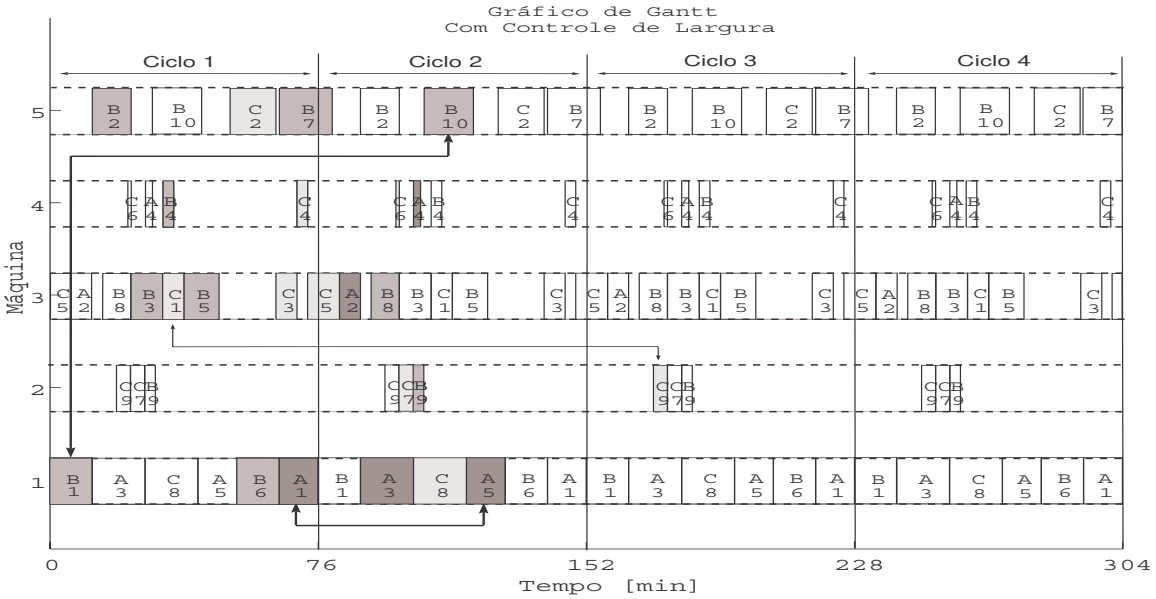



**Figura B.2**: Exemplo de um Job-Shop Cíclico Com Largura $\leqslant 2$

através da equação,

$$t_{j,k+h} \geqslant t_{i,k} + p_i, \quad h \in \mathbb{Z}$$

para qualquer número natural k. Assim a h-ésima recorrência da k-ésima ocorrência da operação j só pode ser inicializada após o término da k-ésima ocorrência de i.

Note que pela definição, a altura de recorrência no job pode ser dada por um número fixo que limita a quantidade de instâncias de um mesmo job que podem ocorrer no mesmo ciclo. Assim, enquanto a largura limita superiormente a quantidade de ciclos utilizados para a finalização de cada iteração de um job, a altura de recorrência de um job limita superiormente a quantidade de instâncias que podem ocorrer num mesmo período. A figura B ilustra melhor esta idéia, mostrando um exemplo com 3 jobs distintos e altura de recorrência igual a 2, e com uma solução factível. Como pode ser notado, cada iteração dos jobs 1, 2 e 3 leva dois ciclos ou menos para ser finalizada, isto corresponde a estabelecer-se uma largura igual ou menor a 2.

Hanen (1994) em seu estudo sobre escalonamento cíclico, propôs uma formulação em programação inteira mista para o problema de escalonamento cíclico com recursos disjuntivos[†] e que denominou job-shop recorrente (JSR). Também Rao (1992) estudou o problema de escalonamento cíclico e propôs uma formulação em programação linear inteira mista (MILP)[‡] para o problema de jobs idênticos visando resolver um problema em um ambiente de manufatura, onde um Conjunto de Peças Minimal (MPS) de jobs é processado infinitas vezes.

O problema de job-shop recorrente (Hanen, 1994) e o problema de escalonamento cíclico de jobs idênticos (Rao, 1992) combinam as restrições do problema de escalonamento básico e máquinas especializadas[§] (*special-purpose*). O objetivo é achar um escalonamento periódico com um

[†]Qualquer seqüência de processamento das tarefas associadas a um recurso é factível.

[‡]mixed integer linear programming

[§]Uma operação qualquer só pode ser processada na máquina associada a ela

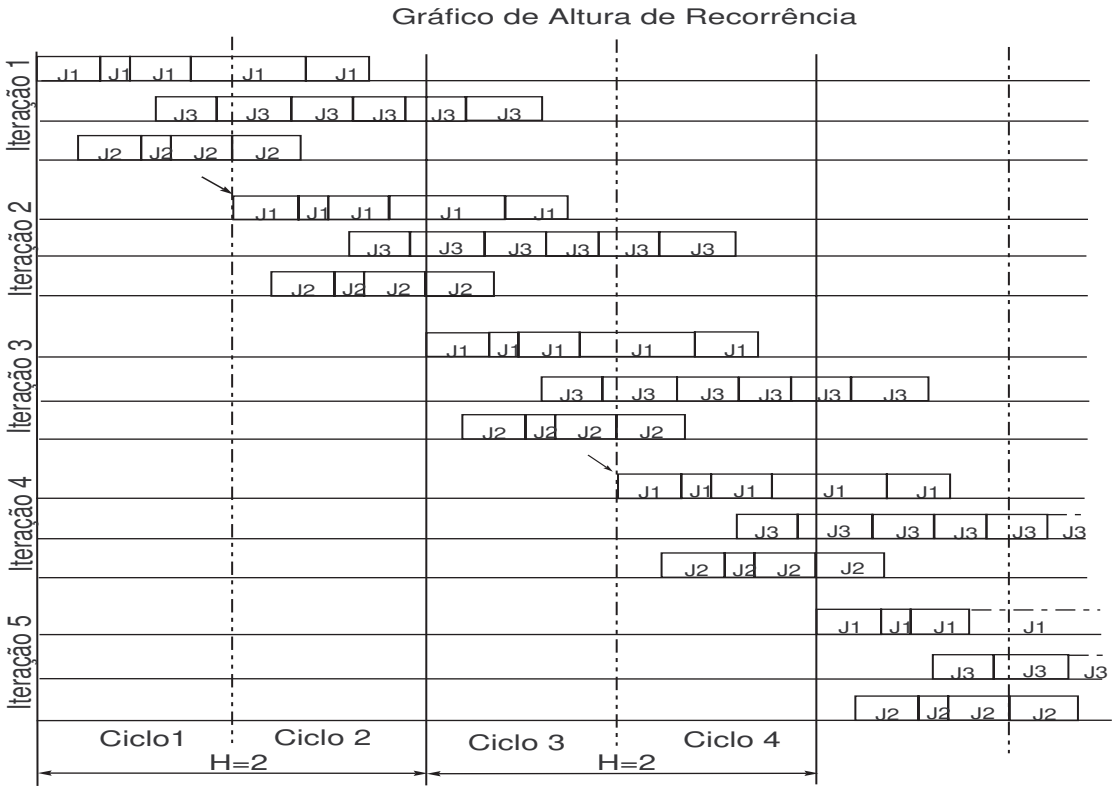



**Figura B.3**: Exemplo de um Job-Shop Cíclico Com Altura de Recorrência igual a 2

fluxo máximo, o que equivale a achar um escalonamento periódico com um período mínimo. Tanto em Rao (1992) quanto em Hanen (1994) os tempos de processamento das operações são fixos, não existindo sobreposição de operações em uma mesma máquina e também cada operação é associada a uma única máquina. Apesar dessa similaridade, os modelos de Hanen (1994) e Rao (1992) não são equivalentes pois cada um otimiza o fluxo de jobs de forma diferente, sem uma correspondência direta. Por esta razão, Portugal (1999) fez modificações nos dois modelos, no intuito de torná-los equivalentes. Estas formulações modificadas foram chamadas, respectivamente por, Job Shop Recorrente Adapatado(JSRA) e Escalonamento Cíclico de Jobs Distintos (ECJD).

As equações que representam as restrições do modelo de JSRA são descritas a seguir:

- Cada job consiste de um conjunto de operações que devem ser realizadas em alguma ordem. Assim estabelecida esta ordem pode-se sugerir um escalonamento cíclico, estabelecendo-se os tempos de inicialização de cada operação, de acordo com esta ordem de precedência; e com a periodicidade surge uma relação de pseudo-precedência entre a primeira e a última operação do job. Pode-se explicitar esta pseudo-precedência determinando que a k-ésima iteração da última operação do job J, seja finalizada até a inicialização da (k+h)-ésima iteração da primeira operação deste job, onde J é um job qualquer e h é um número inteiro. Estas duas relações, precedência e pseudo-precedência, podem ser representadas pela seguinte

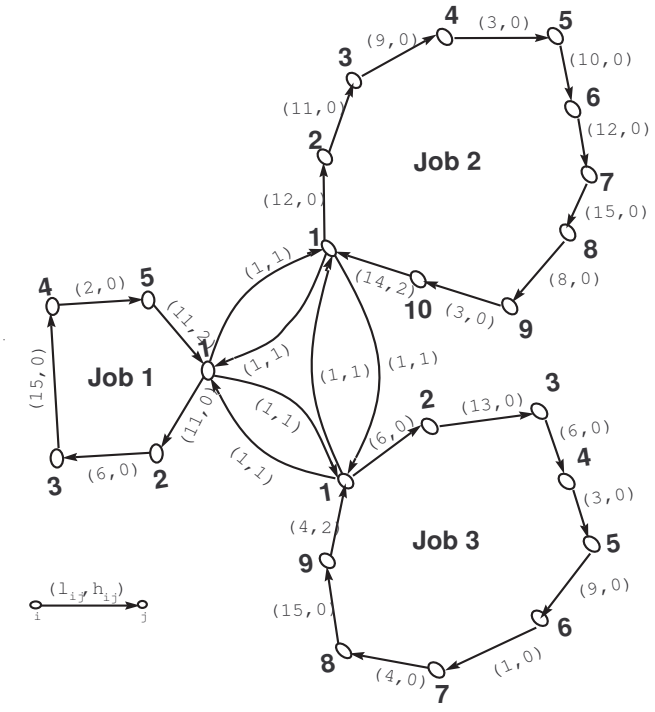



**Figura B.4**: Exemplo de um Job Shop Cíclico

equação:

$$t_{Jj} - t_{Ji} \geqslant p_{Ji} - h_{Ji}z \quad \forall i,j \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}. \tag{B.1}$$

Esta relação de pseudo-precedência é chamada neste trabalho de altura de recorrência do job.

- O conjunto de operações associadas a uma determinada máquina obedece a um certo escalonamento parcial gerado na resolução do problema, porém é necessário garantir a não sobreposição de operações na máquina e para isso as seguintes restrições serão utilizadas:

$$t_{Qj} - t_{Ji} \geqslant p_{Ji} - k(Ji,Qj)z \quad \forall Ji,Qj \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}, \tag{B.2}$$

$$k(Ji,Qj) + k(Qj,Ji) = 1 \quad \forall Ji,Qj \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}, \tag{B.3}$$

$$k(Ji,Qj) \in \mathbb{Z} \quad \forall Ji,Qj \in \mathbb{O}_m, \quad \forall m \in \mathbb{M}. \tag{B.4}$$

- Para se ter um equilíbrio (ou justiça) na produção dos jobs distintos ao longo do tempo e determinar que todos os jobs comecem no mesmo período, estipulou-se uma altura de recorrência entre as primeiras operações dos jobs distintos e que é mostrada a seguir:

$$t_{J1} \geqslant 1 - z \quad \forall J \in \mathbb{J}. \tag{B.5}$$

Como a variável t pode assumir o valor nulo e toda primeira operação de cada *job* deve iniciar no 1º período, estas operações devem ser estritamente menores que o valor do ciclo, dado por z.

- Limites das variáveis:

$$t_{Ji} \geqslant 0 \quad \forall Ji \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}, \tag{B.6}$$

$$z \geqslant 0. \tag{B.7}$$

E o conjunto de restrições do JSRA será então:

$$\begin{aligned}
\min z & \\
t_{Jj} - t_{Ji} &\geqslant p_{Ji} - h_{Ji} z \quad \forall i,j \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}; \\
t_{Qj} - t_{Ji} &\geqslant p_{Ji} - k(Ji,Qj) z, \quad \forall Ji,Qj \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}; \\
k(Ji,Qj) + k(Qj,Ji) &= 1 \quad \forall Ji,Qj \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}; \\
t_{J1} &\geqslant 1 - z, \quad \forall J \in \mathbb{J}; \\
t_{Ji} &\geqslant 0 \quad \forall Ji \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}; \\
z &\geqslant 0; \\
k(Ji,Qj) &\in \mathbb{Z} \quad \forall Ji,Qj \in \mathbb{O}_m, \quad \forall m \in \mathbb{M}.
\end{aligned} \tag{B.8}$$

Baseado no modelo de Rao(1992), o seguinte conjunto de restrições foi proposto para solucionar o Problema de Job-Shop Cíclico, denominado por escalonamento cíclico de jobs distintos (ECJD):

- O conjunto de operações associado a um determinado job deve obedecer a uma certa ordem denominada de restrições de precedência no job:

$$T_{s_{Ji}} - T_{Ji} \geqslant p_{Ji} - O_{Ji} z \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}. \tag{B.9}$$

- O conjunto de operações associadas a uma determinada máquina obedece a um certo escalonamento parcial gerado na resolução do problema, mas para garantir a não sobreposição de operações na máquina a seguinte restrição é utilizada:

$$T_{\sigma_{Ji}} - T_{Ji} \geqslant p_{Ji} - L_{Ji} z \quad \forall Ji \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}, \tag{B.10}$$

$$\sum_{Ji \in \mathbb{O}_s} L_{Ji} = 1 \quad \forall s \in \mathbb{M}. \tag{B.11}$$

- Para garantir a não existência de subciclos nas máquinas as seguintes restrições são utilizadas:

$$v_{Ji} - v_{Qj} + n x_{ij} \leqslant (n-1) + n(L_{Ji} + L_{Qj}) \quad \forall Ji,Qj \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}, \tag{B.12}$$

$$\sum_{Ji \in \mathbb{O}_s} x_{Ji,Qj} = 1 \quad \forall Qj \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}, \tag{B.13}$$

$$\sum_{Qj \in \mathbb{O}_s} x_{Ji,Qj} = 1 \quad \forall Ji \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}. \tag{B.14}$$

- Para se controlar o fluxo do conjunto de jobs, será feito o controle de largura de cada job através da altura de recorrência, assim:

$$T_{J1} - T_{Jn} \geqslant p_{Jn} - O_{Jn} z \quad n = card(\mathbb{O}_J) \quad \forall J \in \mathbb{J}, \tag{B.15}$$

$$\sum_{i=1; i \in \mathbb{O}_J}^{n} O_{Ji} \leqslant H \quad \forall J \in \mathbb{J}. \tag{B.16}$$

$$\sum_{i=1; i \in \mathbb{O}_J}^{n} O_{Ji} \geqslant 1 \quad \forall J \in \mathbb{J}. \tag{B.17}$$

  onde $n = card(J)$ e $H \in \mathbb{N}$.

- Para se ter um equilíbrio na produção dos jobs ao longo do tempo propõe-se uma altura de recorrência entre as primeiras operações dos jobs distintos:

$$T_{J1} \geqslant 1 - z \quad \forall J \in \mathbb{J}. \tag{B.18}$$

- Limites das variáveis:

$$T_{Ji} \geqslant 0, \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}, \tag{B.19}$$

$$T_{Ji} < z, \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}, \tag{B.20}$$

$$O_{Ji} \leqslant 2, \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}, \tag{B.21}$$

$$O_{Ji} \in \mathbb{Z} \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}, \tag{B.22}$$

$$\nu_{Ji} \in \mathbb{N} \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}, \tag{B.23}$$

$$z \geqslant 0. \tag{B.24}$$

  Pela equação B.20 vê-se que a equação B.18 nesta modelagem é redundante e por isso não é necessária incluí-la no conjunto de restrições.

E o conjunto de restrições do ECJD será então:

$$\begin{aligned}
&\min Z \\
T_{s_{Ji}} - T_{Ji} &\geqslant p_{Ji} - O_{Ji} z \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J}; \\
T_{\sigma_{Ji}} - T_{Ji} &\geqslant p_{Ji} - L_{Ji} z \quad \forall Ji \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}; \\
\sum_{Ji \in \mathbb{O}_s} L_{Ji} &= 1 \quad \forall s \in \mathbb{M}; \\
\nu_{Ji} - \nu_{Qj} + n x_{ij} &\leqslant (n-1) + n(L_{Ji} + L_{Qj}) \quad \forall Ji, Qj \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}; \\
\sum_{Ji \in \mathbb{O}_s} x_{Ji,Qj} &= 1 \quad \forall Qj \in \mathbb{O}_s, \quad \forall s \in \mathbb{M}; \\
\sum_{Qj \in \mathbb{O}_s} x_{Ji,Qj} &= 1 \quad \forall Ji \in \mathbb{O}_s, \quad \forall s \in \mathbb{M};
\end{aligned} \tag{B.25}$$

$$
\begin{aligned}
\sum_{i=1;i\in\mathbb{O}_J}^{n} O_{Ji} &\leqslant H \quad \forall J \in \mathbb{J};\\
\sum_{i=1;i\in\mathbb{O}_J}^{n} O_{Ji} &\geqslant 1 \quad \forall J \in \mathbb{J};\\
T_{J1} - T_{Q1} &\geqslant 1 - z \quad \forall J, Q \in \mathbb{J};\\
T_{Ji} &\geqslant 0, \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J};\\
T_{Ji} &< z, \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J};\\
O_{Ji} &\leqslant 2, \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J};\\
O_{Ji} &\in \mathbb{Z} \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J};\\
\nu_{Ji} &\in \mathbb{N} \quad \forall i \in \mathbb{O}_J, \quad \forall J \in \mathbb{J};\\
z &\geqslant 0.
\end{aligned}
$$

# C

# Programas em GAMS

## C.1 Exemplo1

```
*************************************************************************
* ESTE MODELO ACHA UM ESCALONAMENTO CICLICO OTIMO EM UM PROBLEMA DE    *
* SHOP.                                                                 *
*************************************************************************
OPTION LIMROW =0;
OPTION LIMCOL=0;
OPTION SOLPRINT=OFF;
OPTION SYSOUT=OFF;
OPTION OPTCR=0.0007;
OPTION RESLIM=300000;
OPTION ITERLIM=15000000;
*************************************************************************
*                 DECLARACAO DOS CONJUNTOS UTILIZADOS                   *
*************************************************************************
SETS  JOB  JOBS / J1*J4 /
REC  RECURSOS / R1*R4 /
OPER  OPERACOES / O1*O21 /
MAQ  MAQUINAS / M1*M3 /
MAQQP  MAQUINAS PARALELAS / M1*M3 /;

ALIAS(JOB,JOBB);
ALIAS(OPER,OPERR);
ALIAS(OPER,OPERS);
ALIAS(OPER,OPES);
ALIAS(MAQ,MAQQ);
ALIAS(REC,RECC);

*************************************************************************
*          HABILITA A EXECUCAO DE UMA OPERACAO EM UMA MAQUINA           *
*************************************************************************

SET HABMAQ(OPER,MAQ) CONJUNTO DE OPERACOES DE CADA MAQUINA
 / O2.M1, O4.M1, O8.M2, O13.M2, O16.M3, O18.M3 /;
SET MAQP(MAQ,MAQQP) CONJUNTO DE MAQUINAS PARALELAS
```

```
 / M1.M1, M2.M2, M3.M3 /;
SET HABOPE(OPER,OPERR) CONJUNTO DE OPERACOES SUCESSORAS
 / O2.O3,O3.O4,O4.O5,O5.O2,O2.O12,O12.O13,O13.O14,O14.O10,O10.O2,O13.O15,
O15.O16,O16.O17,O17.O18,O18.O19,O19.O2,O21.O8,O8.O9,O9.O10,O10.O21 /;
SET OPERACOES(OPER) CONJUNTO DE OPERACOES REAIS
 / O2,O3,O4,O5,O8,O9,O10,O12,O13,O14,O15,O16,O17,O18,O19,O21 /;
SET HABJOB(JOB,OPER) CONJUNTO DE OPERACOES DE CADA PROCESSO SIMPLES
 / J1.O2,J1.O3,J1.O4,J1.O5,J2.O2,J2.O12,J2.O13,J2.O14,J2.O10,J3.O2,J3.O12
J3.O13,J3.O15,J3.O16,J3.O17,J3.O18,J3.O19,J4.O21,J4.O8,J4.O9,J4.O10 /;
SET JOBPRIM(JOB,OPER) CONJUNTO DE PRIMEIRA OPERACAO DO JOB
 / J1.O2,J2.O2,J3.O2,J4.O21 /;
SET JOBULT(OPER) CONJUNTO DE PRIMEIRA OPERACAO DO JOB
 / O5,O10,O19, /;
SET RECJOB(REC,JOB) CONJUNTO DE JOBS DO PROCESSO
 / R1.J1,R2.J2,R3.J3,R4.J4 /;
SET RECREC(REC,RECC) CONJUNTO DE RECURSOS SINCRONOS
 / R1.R1,R1.R2,R1.R3,R2.R2,R2.R3,R3.R3,R4.R4 /;
SET JOBJOB(JOB,JOBB) CONJUNTO DE RECURSOS SINCRONOS
 / J1.J1,J2.J2,J3.J3,J4.J4,J1.J2,J1.J3,J2.J3,J2.J4 /;
PARAMETER DADOS(OPER) TEMPOS DE DURACAO DAS OPERACOES
 /O2 200,O3 100,O4 200,O5 300,O8 200,O9 100,O10 100
O12 100,O13 200,O14 200,O15 100,O16 200,O17 100,O18 200
O19 300,O21 100 /;
PARAMETER MAQUINA(MAQ)  QUANTIDADE MAQUINAS IDENTICAS
 / M1 1,M2 1,M3 1 /;
PARAMETER ALTURA(JOB)  ALT DE RECORRENCIA DE CADA PROCESSO
 / J1 1,J2 1,J3 1,J4 1  /;
PARAMETER RECURSO(REC)  REC DISP PARA DISTRIBUIR ENTRE OS PROCS PARALELOS
 / R1 1,R2 1,R3 1,R4 1 /;

VARIABLES       TAO                   FLUXO DO SISTEMA
                W                     FLUXO OTIMO
                PER                   PERIODO OTIMO
                TI(OPER)              TEMPO INICIAL DE CADA OPERACAO
                O(JOB,OPER)           TEMPO EXECUCAO DE CADA OPERACAO
                K(OPER,OPERR,MAQ)     PARAMETRO DE MAQUINA
                X(OPER,MAQ,MAQQP)     PARAMETRO DE MAQUINA
                U(OPER)               TEMPO INICIAL MOD PERIODO
POSITIVE VARIABLES TAO,U,TI,PER;
INTEGER VARIABLES D;
BINARY VARIABLES K,X;
SCALAR  B                     DICOTOMIA;
B = 2700;
EQUATIONS
PREC1(JOB,OPER,OPERR)      PRECEDENCIA ENTRE OPERACOES DO  MESMO JOB
PREC1A(REC,RECC)           RECURSOS SINCRONOS
PREC1C(OPER,OPERR,OPERS)   OPERACOES COM MESMA PRECEDENCIA COMECA AO MESMO TEMPO
PREC1D(JOB,OPER,OPERR)     OPERACOES COM MESMA PRECEDENCIA COMECA AO MESMO TEMPO
```

```
MAQU(OPER,OPERR,MAQ,MAQQP) IMPOSSIBILITA SIMULTANEIDADE DE OPERACOES NUMA MAQUINA
KDEC1(OPER,OPERR,MAQ)      NUMERO BINARIO QUE INDICA QUAL OPERACAO EXECUTAR
KDEC3(OPER,MAQ,JOB)        NUMERO BINARIO QUE INDICA QUAL OPERACAO EXECUTAR
LIMINF(JOB,OPER)           LIMITA INFERIORMENTE A VARIAVEL O
LIMSUP(JOB,OPER)           LIMITA SUPERIORMENTE A VARIAVEL O
LIM(OPER)                  IMPOE LIMITES A VARIAVEIS (U=<1-TAO)
TTINICIAL1(JOB)            ESTABELECE A ALTURA DE RECORRENCIA
TTINICIAL2(JOB)            ESTABELECE A ALTURA DE RECORRENCIA
OBJETIVO                   FUNCAO OBJETIVO;

LIM(OPER)$(OPERACOES(OPER))..U(OPER)=L=1-TAO;

KDEC3(OPER,MAQ,JOB)$(HABMAQ(OPER,MAQ)AND HABJOB(JOB,OPER))..
SUM((MAQQP)$(MAQP(MAQ,MAQQP)), X(OPER,MAQ,MAQQP)) =E=  1;

LIMINF(JOB,OPER)$(HABJOB(JOB,OPER))..D(JOB,OPER)=G=0;

LIMSUP(JOB,OPER)$(HABJOB(JOB,OPER))..D(JOB,OPER)=L=2;

TTINICIAL1(JOB)..SUM((OPER)$(HABJOB(JOB,OPER)), D(JOB,OPER)) =L=ALTURA(JOB);

TTINICIAL2(JOB)..SUM((OPER)$(HABJOB(JOB,OPER)), D(JOB,OPER)) =G=1;

PREC1(JOB,OPER,OPERR)$(HABJOB(JOB,OPER) AND HABJOB(JOB,OPERR) AND
HABOPE(OPER,OPERR))..TAO*DADOS(OPER) =L= U(OPERR)-U(OPER)+ D(JOB,OPER);

PREC1A(REC,RECC)$(RECREC(REC,RECC))..SUM((OPER,JOB)$(JOBPRIM(JOB,OPER)AND RECJOB(REC,JOB)),
U(OPER))=E=SUM((OPER,JOBB)$(JOBPRIM(JOBB,OPER)AND RECJOB(RECC,JOBB)), U(OPER));

PREC1C(OPER,OPERR,OPERS)$((ORD(OPER) NE ORD(OPERR))AND HABOPE(OPERS,OPER)AND
HABOPE(OPERS,OPERR)AND NOT(JOBULT(OPERS)))..U(OPERR)=E=U(OPER);

PREC1D(JOB,OPER,OPERR)$(HABJOB(JOB,OPER) AND NOT(HABJOB(JOB,OPERR))AND
HABOPE(OPER,OPERR))..TAO*DADOS(OPER) =L= U(OPERR)-U(OPER)+ D(JOB,OPER);

MAQU(OPER,OPERR,MAQ,MAQQP)$(HABMAQ(OPER,MAQ) AND HABMAQ(OPERR,MAQ)AND MAQP(MAQ,MAQQP)AND
(ORD(OPER) NE ORD(OPERR)))..TAO*DADOS(OPER)=L=U(OPERR)-U(OPER)+K(OPER,OPERR,MAQ) - 2
X(OPER,MAQ,MAQQP) + X(OPERR,MAQ,MAQQP);

KDEC1(OPER,OPERR,MAQ)$(HABMAQ(OPER,MAQ) AND HABMAQ(OPERR,MAQ)AND
(ORD(OPER) LT ORD(OPERR)))..K(OPER,OPERR,MAQ)+K(OPERR,OPER,MAQ)=E=1;
OBJETIVO.. W=E=TAO;

MODEL  CICLICO /ALL/;
CICLICO.WORKSPACE=30;
SOLVE     CICLICO USING MIP MAXIMIZING W;

PARAMETER TINICIAL(JOB,OPER,MAQ)     CALCULA O TEMPO INICIAL DE CADA OPERACAO;
```

```
TI.L(OPER)=(U.L(OPER)/TAO.L)$(TAO.L GT 0);
PARAMETER PERIODO     CALCULA A DURACAO DO PERIODO;
PER.L=(1/TAO.L)$(TAO.L GT 0);
DISPLAY TI.L,PER.L,D.L,U.L,TAO.L;
```

### C.1.1 Resoluçao

A seguir será apresentada a solução deste problema gerada pelo GAMS, porém como a saída do GAMS inclui a formulação apresentada anteriormente, apenas a parte relativa a solução será apresentada:

```
COMPILATION TIME     =        0.000 SECONDS    0.7 Mb      WIN194-116
PJSC                                                 05/21/06 11:49:19  PAGE      8
Model Statistics    SOLVE CICLICO USING MIP FROM LINE 287   GAMS Rev 116  Windows NT/95/98

MODEL STATISTICS

BLOCKS OF EQUATIONS      13     SINGLE EQUATIONS      117
BLOCKS OF VARIABLES       6     SINGLE VARIABLES       51
NON ZERO ELEMENTS       289     DISCRETE VARIABLES     33

GENERATION TIME      =        0.015 SECONDS    1.4 Mb      WIN194-116

EXECUTION TIME       =        0.015 SECONDS    1.4 Mb      WIN194-116
PJSC                                          05/21/06 11:49:19  PAGE      9
                                                GAMS Rev 116  Windows NT/95/98

               S O L V E      S U M M A R Y

     MODEL   CICLICO             OBJECTIVE  W
     TYPE    MIP                 DIRECTION  MAXIMIZE
     SOLVER  OSL                 FROM LINE  287

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE                0.0007

 RESOURCE USAGE, LIMIT          0.047   300000.000
 ITERATION COUNT, LIMIT        40     15000000

 OSL Version 1 Aug  7, 2000 WIN.OS.OS 19.4 058.040.038.WAT
 Work space requested by user   --   30.00 Mb
 Work space requested by solver --    0.25 Mb

**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
```

```
PJSC                                                 05/21/06 11:49:19  PAGE     10
E x e c u t i o n                                   GAMS Rev 116  Windows NT/95/98

----    341 VARIABLE  TI.L  TEMPO INICIAL DE CADA OPERACAO
O2  1200.000,    O4   100.000,    O5   300.000,    O8  1200.000,    O10 1000.000
O13  100.000,    O14  300.000,    O15  300.000,    O16  400.000,    O17  600.000
O18  700.000,    O19  900.000,    O21 1100.000
----    341 VARIABLE  PER.L               =     1400.000  PERIODO OTIMO
----    341 VARIABLE  D.L
            O2          O8
J1       1.000
J2       1.000
J3       1.000
J4                  1.000
----    341 VARIABLE  U.L  TEMPO INICIAL MOD PERIODO
O2  0.857,    O4  0.071,    O5  0.214,    O8  0.857,    O10 0.714,    O13 0.071
O14 0.214,    O15 0.214,    O16 0.286,    O17 0.429,    O18 0.500,    O19 0.643
O21 0.786
----    341 VARIABLE  TAO.L                =  7.142857E-4  FLUXO DO SISTEMA
**** REPORT FILE SUMMARY
TESEA C:\USUARIOS\DENISE\ALGOTESE\TESEA.PUT
TESEB C:\USUARIOS\DENISE\ALGOTESE\TESEB.PUT
TESEC C:\USUARIOS\DENISE\ALGOTESE\TESEC.PUT
TEMPOS C:\USUARIOS\DENISE\ALGOTESE\TEMPOS.PUT
ALTREL C:\USUARIOS\DENISE\ALGOTESE\ALTREL.PUT

EXECUTION TIME       =        0.016 SECONDS    1.4 Mb      WIN194-116

USER: Leo Pini Magalhaes                             G000925:1529AP-WIN
      UNICAMP, FEE/DCA                                           DC2914
**** FILE SUMMARY
INPUT      C:\USUARIOS\DENISE\ALGOTESE\MODELO3.GMS
OUTPUT     C:\USUARIOS\DENISE\ALGOTESE\MODELO3.LST
```

## C.2 Exemplo2

Este programa corresponde ao exemplo 6.2 do capítulo 6.

```
*************************************************************************
* ESTE MODELO ACHA UM ESCALONAMENTO CICLICO OTIMO EM UM PROBLEMA DE    *
* SHOP.                                                                 *
*************************************************************************
OPTION LIMROW =0;
OPTION LIMCOL=0;
OPTION SOLPRINT=OFF;
OPTION SYSOUT=OFF;
OPTION OPTCR=0.0007;
```

```
OPTION RESLIM=300000;
OPTION ITERLIM=15000000;
************************************************************************
*                 DECLARACAO DOS CONJUNTOS UTILIZADOS                  *
************************************************************************
SETS  JOB  JOBS / J1*J8 /
REC  RECURSOS / R1*R2 /
OPER  OPERACOES / O1*O47 /
MAQ  MAQUINAS / M1*M4 /
MAQQP  MAQUINAS PARALELAS / M1*M4 /;
ALIAS(JOB,JOBB);
ALIAS(OPER,OPERR);
ALIAS(OPER,OPERS);
ALIAS(OPER,OPES);
ALIAS(MAQ,MAQQ);
ALIAS(REC,RECC);
************************************************************************
*          HABILITA A EXECUCAO DE UMA OPERACAO EM UMA MAQUINA          *
************************************************************************
SET HABMAQ(OPER,MAQ) CONJUNTO DE OPERACOES DE CADA MAQUINA
 / O9.M1,O28.M1,O11.M1,O30.M1,O12.M2,O31.M2,O15.M2,O38.M2,O10.M3,O29.M3
O17.M3,O40.M3,O16.M4,O39.M4,O18.M4,O41.M4 /;
SET MAQP(MAQ,MAQQP) CONJUNTO DE MAQUINAS PARALELAS
 / M1.M1,M2.M2,M3.M3,M4.M4 /;
SET HABOPE(OPER,OPERR) CONJUNTO DE OPERACOES SUCESSORAS
 / O5.O9,O9.O13,O13.O15,O15.O19,O19.O5,O22.O10,O10.O32,O32.O38,O38.O42
O42.O22,O23.O28,O28.O33,O33.O16,O16.O43,O43.O23,O24.O29,O29.O34,O34.O39
O39.O44,O44.O24,O6.O11,O11.O14,O14.O17,O17.O20,O20.O6,O25.O12,O12.O35
O35.O40,O40.O45,O45.O25,O26.O30,O30.O36,O36.O18,O18.O46,O46.O26,O27.O31
O31.O37,O37.O41,O41.O47,O47.O27  /;
SET OPERACOES(OPER) CONJUNTO DE OPERACOES REAIS
 / O5,O22,O23,O24,O6,O25,O26,O27,O9,O28,O10,O29,O11,O30,O12,O31,O13,O32
O33,O34,O14,O35,O36,O37,O15,O38,O16,O39,O17,O40,O18,O41,O19,O42,O43,O44
O20,O45,O46,O47 /;
SET HABJOB(JOB,OPER) CONJUNTO DE OPERACOES DE CADA PROCESSO SIMPLES
 / J1.O5,J1.O9,J1.O13,J1.O15,J1.O19,J2.O22,J2.O10,J2.O32,J2.O38,J2.O42
J3.O23,J3.O28,J3.O33,J3.O16,J3.O43,J4.O24,J4.O29,J4.O34,J4.O39,J4.O44
J5.O6,J5.O11,J5.O14,J5.O17,J5.O20,J6.O25,J6.O12,J6.O35,J6.O40,J6.O45
J7.O26,J7.O30,J7.O36,J7.O18,J7.O46,J8.O27,J8.O31,J8.O37,J8.O41,J8.O47 /;
SET JOBPRIM(JOB,OPER) CONJUNTO DE PRIMEIRA OPERACAO DO JOB
 / J1.O5,J2.O22,J3.O23,J4.O24,J5.O6,J6.O25,J7.O26,J8.O27 /;
SET JOBULT(OPER) CONJUNTO DE PRIMEIRA OPERACAO DO JOB
 / O19,O42,O43,O44,O20,O45,O46,O47  /;
SET RECJOB(REC,JOB) CONJUNTO DE JOBS DO PROCESSO
 / R1.J1,R1.J2,R1.J3,R1.J4,R2.J5,R2.J6,R2.J7,R2.J8 /;
SET RECREC(REC,RECC) CONJUNTO DE RECURSOS SINCRONOS
 / R1.R1,R1.R2,R2.R2 /;
SET JOBJOB(JOB,JOBB) CONJUNTO DE RECURSOS SINCRONOS
```

```
 / J1.J1,J2.J2,J3.J3,J4.J4,J5.J5,J6.J6,J7.J7,J8.J8 /;

PARAMETER DADOS(OPER) TEMPOS DE DURACAO DAS OPERACOES
 / O5 12,O22 12,O23 12,O24 12,O6 13,O25 13,O26 13,O27 13,O9 34,O28 34
O10 43,O29 43,O11 2,O30 2,O12 33,O31 33,O13 66,O32 66,O33 66,O34 66
O14 12,O35 12,O36 12,O37 12,O15 12,O38 12,O16 10,O39 10,O17 20,O40 20
O18 22,O41 22,O19 12,O42 12,O43 12,O44 12,O20 21,O45 21,O46 21,O47 21 /;
PARAMETER MAQUINA(MAQ)  QUANTIDADE MAQUINAS IDENTICAS
 / M1 1,M2 1,M3 1,M4 1 /;
PARAMETER ALTURA(JOB)  ALT DE RECORRENCIA DE CADA PROCESSO
 / J1 1,J2 1,J3 1,J4 1,J5 1,J6 1,J7 1,J8 1 /;
PARAMETER RECURSO(REC)  REC DISP PARA DISTRIBUIR ENTRE OS PROCS PARALELOS
 / R1 4,R2 4 /;

VARIABLES       TAO                    FLUXO DO SISTEMA
                W                      FLUXO OTIMO
                PER                    PERIODO OTIMO
                TI(OPER)               TEMPO INICIAL DE CADA OPERACAO
                O(JOB,OPER)            TEMPO EXECUCAO DE CADA OPERACAO
                K(OPER,OPERR,MAQ)      PARAMETRO DE MAQUINA
                X(OPER,MAQ,MAQQP)       PARAMETRO DE MAQUINA
                U(OPER)                TEMPO INICIAL MOD PERIODO
                C(JOB)             PARAMETRO DO PROCESSO
POSITIVE VARIABLES TAO,U,TI,PER;
INTEGER VARIABLES D;
BINARY VARIABLES K,C,X;
SCALAR  B                     DICOTOMIA;
B = 312;

EQUATIONS
PREC1(JOB,OPER,OPERR)      PRECEDENCIA ENTRE OPERACOES DO  MESMO JOB
PREC1A(REC,RECC)           RECURSOS  SINCRONOS
PREC1C(OPER,OPERR,OPERS)   OPERACOES COM MESMA PRECEDENCIA COMECA AO MESMO TEMPO
PREC1D(JOB,JOBB,OPER,OPERR)     OPERACOES DE RECURSOS DIFERENTES COM MESMA PRECEDENCIA
PREC1B(REC,RECC)           DEVEM COMECAR AO MESMO TEMPO
PREC2(REC)                 DETERMINA QUE PELO MENOS UM JOB SEJA EXECUTADO
PREC3(REC)                 LIMITA A QUANTIDADE DE JOBS IDENTICOS QUE PODEM SER EXECUTADOS
MAQU(OPER,OPERR,MAQ,MAQQP,JOB,JOBB) IMPOSSIBILITA SIMULTANEIDADE DE OPERACOES NUMA MAQUINA
KDEC1(OPER,OPERR,MAQ)      NUMERO BINARIO QUE INDICA QUAL OPERACAO EXECUTAR
KDEC3(OPER,MAQ)        NUMERO BINARIO QUE INDICA QUAL OPERACAO EXECUTAR
LIMINF(JOB,OPER)           LIMITA INFERIORMENTE A VARIAVEL O
LIMSUP(JOB,OPER)           LIMITA SUPERIORMENTE A VARIAVEL O
LIM(OPER)                  IMPOE LIMITES A VARIAVEIS (U=<1-TAO)
TTINICIAL1(JOB)            ESTABELECE A ALTURA DE RECORRENCIA
TTINICIAL2(JOB)            ESTABELECE A ALTURA DE RECORRENCIA
TTINICIAL3(JOB)
OBJETIVO                   FUNCAO OBJETIVO;
```

```
LIM(OPER)$OPERACOES(OPER)..U(OPER)=L=1-TAO;

PREC2(REC)..SUM((JOB)$(RECJOB(REC,JOB)), C(JOB)) =G=1;

PREC3(REC)..SUM((JOB)$(RECJOB(REC,JOB)), C(JOB)) =L=RECURSO(REC);

KDEC3(OPER,MAQ)$(HABMAQ(OPER,MAQ))..SUM((MAQQP)$(MAQP(MAQ,MAQQP)), X(OPER,MAQ,MAQQP)) =E=  1;

LIMINF(JOB,OPER)$(HABJOB(JOB,OPER))..D(JOB,OPER)=G=0;

LIMSUP(JOB,OPER)$(HABJOB(JOB,OPER))..D(JOB,OPER)=L=2;

TTINICIAL1(JOB)..SUM((OPER)$(HABJOB(JOB,OPER)), D(JOB,OPER)) =L=ALTURA(JOB);

TTINICIAL2(JOB)..SUM((OPER)$(HABJOB(JOB,OPER)), D(JOB,OPER)) =G=C(JOB);

TTINICIAL3(JOB)..SUM((OPER)$(HABJOB(JOB,OPER)), U(OPER))=L=B*C(JOB);

PREC1A(REC,RECC)$(RECREC(REC,RECC))..SUM((OPER,JOB)$(JOBPRIM(JOB,OPER)AND RECJOB(REC,JOB)),
U(OPER))=E=SUM((OPER,JOBB)$(JOBPRIM(JOBB,OPER)AND RECJOB(RECC,JOBB)), U(OPER));

PREC1C(OPER,OPERR,OPERS)$((ORD(OPER) NE ORD(OPERR))AND HABOPE(OPERS,OPER)AND
HABOPE(OPERS,OPERR)AND JOBULT(OPERS))..U(OPERR)=E=U(OPER);

PREC1D(JOB,JOBB,OPER,OPERR)$(HABJOB(JOB,OPER) AND NOT(HABJOB(JOBB,OPERR))AND
HABOPE(OPER,OPERR)AND (NOT(JOBJOB(JOB,JOBB)))
AND (NOT(JOBJOB(JOBB,JOB))))..TAO*DADOS(OPER) =L= U(OPERR)-U(OPER)+ D(JOB,OPER)+ 1 - C(JOB);

PREC1B(REC,RECC)$(RECREC(REC,RECC))..SUM((JOB)$(RECJOB(REC,JOB)),
C(JOB))=E=SUM((JOB)$(RECJOB(RECC,JOB)), C(JOB));

PREC1(JOB,OPER,OPERR)$(HABJOB(JOB,OPER) AND HABJOB(JOB,OPERR) AND HABOPE(OPER,OPERR))
..TAO*DADOS(OPER) =L= U(OPERR)-U(OPER)+ D(JOB,OPER) + 1 - C(JOB);

MAQU(OPER,OPERR,MAQ,MAQQP,JOB,JOBB)$(HABMAQ(OPER,MAQ) AND HABMAQ(OPERR,MAQ)AND MAQP(MAQ,MAQQP)AND
(ORD(OPER) NE ORD(OPERR))AND HABJOB(JOB,OPER) AND HABJOB(JOBB,OPERR))..
TAO*DADOS(OPER)+U(OPER)=L=U(OPERR)+K(OPER,OPERR,MAQ) + 4 - X(OPER,MAQ,MAQQP) - X(OPERR,MAQ,MAQQP)
- C(JOB) -C(JOBB);

KDEC1(OPER,OPERR,MAQ)$(HABMAQ(OPER,MAQ) AND HABMAQ(OPERR,MAQ)AND
(ORD(OPER) LT ORD(OPERR)))..K(OPER,OPERR,MAQ)+K(OPERR,OPER,MAQ)=E=1;

OBJETIVO.. W=E=TAO;

MODEL  CICLICO /ALL/;
CICLICO.WORKSPACE=30;

SOLVE     CICLICO USING MIP MAXIMIZING W;
```

```
PARAMETER TINICIAL(JOB,OPER,MAQ)     CALCULA O TEMPO INICIAL DE CADA OPERACAO;
TI.L(OPER)=(U.L(OPER)/TAO.L)$(TAO.L GT 0);
PARAMETER PERIODO     CALCULA A DURACAO DO PERIODO;
PER.L=(1/TAO.L)$(TAO.L GT 0);

DISPLAY TI.L,PER.L,C.L,D.L,U.L,TAO.L;
```

## C.2.1 Resoluçao

A seguir será apresentada a solução deste problema gerada pelo GAMS, porém como a saída do GAMS inclui a formulação apresentada anteriormente, apenas a parte relativa a solução será apresentada:

```
COMPILATION TIME     =        0.031 SECONDS    0.7 Mb      WIN194-116
PJSC                                                 05/21/06 11:48:02  PAGE     10
Model Statistics    SOLVE CICLICO USING MIP FROM LINE 412       GAMS Rev 116  Windows NT/95/98

MODEL STATISTICS

BLOCKS OF EQUATIONS      17     SINGLE EQUATIONS      559
BLOCKS OF VARIABLES       7     SINGLE VARIABLES      154
NON ZERO ELEMENTS      2378     DISCRETE VARIABLES    112

GENERATION TIME      =        0.031 SECONDS    1.4 Mb      WIN194-116
EXECUTION TIME       =        0.062 SECONDS    1.4 Mb      WIN194-116
PJSC                                                 05/21/06 11:48:02  PAGE     11
                                                      GAMS Rev 116  Windows NT/95/98

               S O L V E      S U M M A R Y
     MODEL   CICLICO             OBJECTIVE  W
     TYPE    MIP                 DIRECTION  MAXIMIZE
     SOLVER  OSL                 FROM LINE  412
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE                0.0075

 RESOURCE USAGE, LIMIT          0.539   300000.000
 ITERATION COUNT, LIMIT       526     15000000

 OSL Version 1 Aug  7, 2000 WIN.OS.OS 19.4 058.040.038.WAT
 Work space requested by user   --   30.00 Mb
 Work space requested by solver --    0.76 Mb

**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
```

```
PJSC                                        05/21/06 11:48:02  PAGE     12
E x e c u t i o n                     GAMS Rev 116  Windows NT/95/98

----    466 VARIABLE  TI.L  TEMPO INICIAL DE CADA OPERACAO
O6   87.000,  O14   2.000,  O16  65.000,  O17  14.000, O20  34.000
O23  87.000,  O28  99.000,  O33 133.000,  O43  75.000

----    466 VARIABLE  PER.L                =      134.000  PERIODO OTIMO
----    466 VARIABLE  C.L  PARAMETRO DO PROCESSO
J3 1.000,    J5 1.000
----    466 VARIABLE  D.L
            O6         O12         O13         O32         O33         O34

J1                              1.000
J2                                          1.000
J3                                                      1.000
J4                                                                  1.000
J5       1.000
J6                  1.000

----    466 VARIABLE  U.L  TEMPO INICIAL MOD PERIODO
O6  0.649,    O14 0.015,    O16 0.485,    O17 0.104,    O20 0.254
O23 0.649,    O28 0.739,    O33 0.993,    O43 0.560
----    466 VARIABLE  TAO.L                =        0.007  FLUXO DO SISTEMA
**** REPORT FILE SUMMARY
TESEA C:\USUARIOS\DENISE\ALGOTESE\TESEA.PUT
TESEB C:\USUARIOS\DENISE\ALGOTESE\TESEB.PUT
TESEC C:\USUARIOS\DENISE\ALGOTESE\TESEC.PUT
TEMPOS C:\USUARIOS\DENISE\ALGOTESE\TEMPOS.PUT
ALTREL C:\USUARIOS\DENISE\ALGOTESE\ALTREL.PUT
EXECUTION TIME       =        0.062 SECONDS    1.4 Mb      WIN194-116
USER: Leo Pini Magalhaes                             G000925:1529AP-WIN
      UNICAMP, FEE/DCA                                           DC2914
**** FILE SUMMARY
INPUT      C:\USUARIOS\DENISE\ALGOTESE\MODELO3.GMS
OUTPUT     C:\USUARIOS\DENISE\ALGOTESE\MODELO3.LST
```

## C.3 Aplicação Industrial

Este programa corresponde ao escalonamento do sistema de manufatura flexível do capítulo 6.3.

```
*************************************************************************
* ESTE MODELO ACHA UM ESCALONAMENTO CICLICO OTIMO EM UM PROBLEMA DE    *
* SHOP.                                                                 *
*************************************************************************
OPTION LIMROW =0;
OPTION LIMCOL=0;
```

```
OPTION SOLPRINT=OFF;
OPTION SYSOUT=OFF;
OPTION OPTCR=0.0007;
OPTION RESLIM=300000;
OPTION ITERLIM=15000000;
************************************************************************
*                DECLARACAO DOS CONJUNTOS UTILIZADOS                   *
************************************************************************
SETS  JOB  JOBS / J1*J24 /
REC  RECURSOS / R1*R5 /
OPER  OPERACOES / O1*O257 /
MAQ  MAQUINAS / M1*M11 /
MAQQP  MAQUINAS PARALELAS / M1*M29 /;
ALIAS(JOB,JOBB);
ALIAS(OPER,OPERR);
ALIAS(OPER,OPERS);
ALIAS(OPER,OPES);
ALIAS(MAQ,MAQQ);
ALIAS(REC,RECC);
************************************************************************
*          HABILITA A EXECUCAO DE UMA OPERACAO EM UMA MAQUINA          *
************************************************************************
SET HABMAQ(OPER,MAQ) CONJUNTO DE OPERACOES DE CADA MAQUINA
 / O4.M1,O81.M1,O82.M1,O83.M1,O84.M1,O85.M1,O86.M1,O87.M1,O8.M1,O88.M1,O89.M1
O90.M1,O11.M1,O91.M1,O92.M1,O93.M1,O66.M1,O158.M1,O159.M1,O160.M1,O69.M1,O163.M1
O164.M1,O165.M1,O4.M2,O81.M2,O82.M2,O83.M2,O84.M2,O85.M2,O86.M2,O87.M2,O20.M2
O100.M2,O21.M2,O101.M2,O22.M2,O102.M2,O23.M2,O103.M2,O53.M2,O144.M2,O56.M2
O147.M2,O66.M2,O158.M2,O159.M2,O160.M2,O69.M2,O163.M2,O164.M2,O165.M2,O71.M2
O166.M2,O72.M2,O167.M2,O8.M3,O88.M3,O89.M3,O90.M3,O11.M3,O91.M3,O92.M3,O93.M3
O20.M3,O100.M3,O21.M3,O101.M3,O22.M3,O102.M3,O23.M3,O103.M3,O20.M4,O100.M4,O21.M4
O101.M4,O22.M4,O102.M4,O23.M4,O103.M4,O38.M4,O119.M4,O40.M4,O120.M4,O44.M4,O127.M4
O45.M4,O128.M4,O53.M4,O144.M4,O54.M4,O145.M4,O55.M4,O146.M4,O56.M4,O147.M4,O71.M4
O166.M4,O72.M4,O167.M4,O75.M4,O168.M4,O76.M4,O169.M4,O20.M5,O100.M5,O38.M5,O119.M5
O53.M5,O144.M5,O54.M5,O145.M5,O21.M6,O101.M6,O22.M6,O102.M6,O25.M6,O107.M6,O26.M6
O108.M6,O38.M6,O119.M6,O40.M6,O120.M6,O44.M6,O127.M6,O45.M6,O128.M6,O54.M6,O145.M6
O55.M6,O146.M6,O58.M6,O151.M6,O59.M6,O152.M6,O71.M6,O166.M6,O72.M6,O167.M6,O75.M6
O168.M6,O76.M6,O169.M6,O23.M7,O103.M7,O40.M7,O120.M7,O55.M7,O146.M7,O56.M7,O147.M7
O21.M8,O101.M8,O22.M8,O102.M8,O33.M8,O114.M8,O34.M8,O115.M8,O54.M8,O145.M8,O55.M8
O146.M8,O67.M8,O161.M8,O68.M8,O162.M8,O36.M9,O116.M9,O117.M9,O118.M9,O42.M9,O121.M9
O122.M9,O123.M9,O47.M9,O132.M9,O133.M9,O134.M9,O50.M9,O135.M9,O136.M9,O137.M9,O21.M10
O101.M10,O22.M10,O102.M10,O29.M10,O112.M10,O30.M10,O113.M10,O38.M10,O119.M10,O40.M10
O120.M10,O54.M10,O145.M10,O55.M10,O146.M10,O63.M10,O156.M10,O64.M10,,O157.M10
O71.M10,O166.M10,O72.M10,O167.M10,O53.M11,O144.M11,O56.M11,O147.M11,O66.M11,O158.M11,
O159.M11,O160.M11,O69.M11,O163.M11,O164.M11,O165.M11,O71.M11,O166.M11,O72.M11,O167.M11 /;

SET MAQP(MAQ,MAQQP) CONJUNTO DE MAQUINAS PARALELAS
 / M1.M1,M2.M2,M3.M3,M4.M4,M5.M5,M6.M6,M7.M7,M8.M8,M8.M12,M8.M13,M8.M14,M8.M15,M8.M16,
M8.M17,M8.M18,M8.M19,M8.M20,M8.M21,M8.M22,M8.M23,M8.M24,M8.M25,M8.M26,M8.M27,M8.M28,
```

```
M8.M29,M9.M9,M10.M10,M11.M11 /;

SET HABOPE(OPER,OPERR) CONJUNTO DE OPERACOES SUCESSORAS
 /O4.O8,O8.O12,O12.O20,O20.O24,O24.O36,O36.O43,O43.O47,O47.O51,O51.O53,O53.O57,O57.O66,
O66.O4,O81.O11,O11.O13,O13.O22,O22.O26,O26.O30,O30.O34,O34.O40,O40.O45,O45.O27,O27.O42,
O42.O46,O46.O50,O50.O52,O52.O55,O55.O59,O59.O64,O64.O68,O68.O72,O72.O76,O76.O61,O61.O69,
O69.O81,O82.O88,O88.O94,O94.O21,O21.O25,O25.O29,O29.O33,O33.O38,O38.O44,O44.O104,
O104.O116,O116.O124,O124.O132,O132.O138,O138.O144,O144.O148,O148.O158,O158.O82,
O83.O89,O89.O95,O95.O100,O100.O105,O105.O117,O117.O125,O125.O133,O133.O139,O139.O54,
O54.O58,O58.O63,O63.O67,O67.O71,O71.O75,O75.O149,O149.O159,O159.O83,O84.O91,O91.O97,
O97.O23,O23.O109,O109.O121,O121.O129,O129.O135,O135.O141,O141.O146,O146.O152,O152.O157,
O157.O162,O162.O167,O167.O169,O169.O153,O153.O163,O163.O84,O85.O92,O92.O98,O98.O102,
O102.O108,O108.O113,O113.O115,O115.O120,O120.O128,O128.O110,O110.O122,O122.O130,O130.O136,
O136.O142,O142.O56,O56.O154,O154.O164,O164.O85,O86.O90,O90.O96,O96.O101,O101.O107,
O107.O112,O112.O114,O114.O119,O119.O127,O127.O106,O106.O118,O118.O126,O126.O134,O134.O140,
O140.O145,O145.O151,O151.O156,O156.O161,O161.O166,O166.O168,O168.O150,O150.O160,O160.O86,
O87.O93,O93.O99,O99.O103,O103.O111,O111.O123,O123.O131,O131.O137,O137.O143,O143.O147,
O147.O155,O155.O165,O165.O87,O66.O8,O158.O88,O159.O89,O160.O90,O69.O11,O163.O91,O164.O92,
O165.O93,O47.O36,O132.O116,O133.O117,O134.O118,O50.O42,O135.O121,O136.O122,O137.O123 /;

SET OPERACOES(OPER) CONJUNTO DE OPERACOES REAIS
 / O4,O81,O82,O83,O84,O85,O86,O87,O8,O88,O89,O90,O11,O91,O92,O93,O12,O94,O95,O96,O13,O97
O98,O99,O20,O100,O21,O101,O22,O102,O23,O103,O24,O104,O105,O106,O25,O107,O26,O108,O27
O109,O110,O111,O29,O112,O30,O113,O33,O114,O34,O115,O36,O116,O117,O118,O38,O119,O40,O120
O42,O121,O122,O123,O43,O124,O125,O126,O44,O127,O45,O128,O46,O129,O130,O131,O47,O132,O133
O134,O50,O135,O136,O137,O51,O138,O139,O140,O52,O141,O142,O143,O53,O144,O54,O145,O55,O146
O56,O147,O57,O148,O149,O150,O58,O151,O59,O152,O61,O153,O154,O155,O63,O156,O64,O157,O66
O158,O159,O160,O67,O161,O68,O162,O69,O163,O164,O165,O71,O166,O72,O167,O75,O168,O76,O169, /;

SET HABOPESEQ(OPER,OPERR,MAQ) SEQ DE OPERACOES SUCESSORAS
 / O4.O8.M1,O81.O11.M1,O82.O88.M1,O83.O89.M1,O84.O91.M1,O85.O92.M1,O86.O90.M1,O87.O93.M1
O4.O20.M2,O81.O22.M2,O82.O21.M2,O83.O100.M2,O84.O23.M2,O85.O102.M2,O86.O101.M2,O87.O103.M2
O53.O66.M2,O144.O158.M2,O56.O164.M2,O147.O165.M2,O71.O159.M2,O166.O160.M2,O72.O69.M2
O167.O163.M2,O8.O20.M3,O88.O21.M3,O89.O100.M3,O90.O101.M3,O11.O22.M3,O91.O23.M3,O92.O102.M3
O93.O103.M3,O38.O44.M4,O119.O127.M4,O40.O45.M4,O120.O128.M4,O71.O75.M4,O166.O168.M4,
O72.O76.M4,O167.O169.M4,O20.O53.M5,O100.O54.M5,O38.O144.M5,O119.O145.M5,O21.O25.M6,
O101.O107.M6,O22.O26.M6,O102.O108.M6,O54.O58.M6,O145.O151.M6,O55.O59.M6,O146.O152.M6
O23.O146.M7,O103.O147.M7,O40.O55.M7,O120.O56.M7,O21.O33.M8,O101.O114.M8,O22.O34.M8
O102.O115.M8,O54.O67.M8,O145.O161.M8,O55.O68.M8,O146.O162.M8,O21.O29.M10,O101.O112.M10
O22.O30.M10,O102.O113.M10,O54.O63.M10,O145.O156.M10,O55.O64.M10,O146.O157.M10,O66.O66.M1,
O158.O158.M1,O159.O159.M1,O160.O160.M1,O69.O69.M1,O163.O163.M1,O164.O164.M1,O165.O165.M1,
O20.O20.M4,O100.O100.M4,O21.O21.M4,O101.O101.M4,O22.O22.M4,O102.O102.M4,O23.O23.M4,
O103.O103.M4,O53.O53.M4,O144.O144.M4,O54.O54.M4,O145.O145.M4,O55.O55.M4,O146.O146.M4,
O56.O56.M4,O147.O147.M4,O36.O36.M9,O116.O116.M9,O117.O117.M9,O118.O118.M9,O42.O42.M9,
O121.O121.M9,O122.O122.M9,O123.O123.M9,O47.O47.M9,O132.O132.M9,O133.O133.M9,O134.O134.M9,
O50.O50.M9,O135.O135.M9,O136.O136.M9,O137.O137.M9,O38.O38.M10,O119.O119.M10,O40.O40.M10,
O120.O120.M10,O71.O71.M10,O166.O166.M10,O72.O72.M10,O167.O167.M10 /;
```

```
SET HABJOB(JOB,OPER) CONJUNTO DE OPERACOES DE CADA PROCESSO SIMPLES
 / J1.O4,J1.O8,J1.O12,J1.O20,J1.O24,J1.O36,J1.O43,J1.O47,J1.O51,J1.O53,J1.O57,J1.O66,
J2.O81,J2.O11,J2.O13,J2.O22,J2.O26,J2.O30,J2.O34,J2.O40,J2.O45,J2.O27,J2.O42,J2.O46,
J2.O50,J2.O52,J2.O55,J2.O59,J2.O64,J2.O68,J2.O72,J2.O76,J2.O61,J2.O69,J3.O82,J3.O88,
J3.O94,J3.O21,J3.O25,J3.O29,J3.O33,J3.O38,J3.O44,J3.O104,J3.O116,J3.O124,J3.O132,J3.O138,
J3.O144,J3.O148,J3.O158,J4.O83,J4.O89,J4.O95,J4.O100,J4.O105,J4.O117,J4.O125,J4.O133,
J4.O139,J4.O54,J4.O58,J4.O63,J4.O67,J4.O71,J4.O75,J4.O149,J4.O159,J5.O84,J5.O91,J5.O97,
J5.O23,J5.O109,J5.O121,J5.O129,J5.O135,J5.O141,J5.O146,J5.O152,J5.O157,J5.O162,J5.O167,
J5.O169,J5.O153,J5.O163,J6.O85,J6.O92,J6.O98,J6.O102,J6.O108,J6.O113,J6.O115,J6.O120,
J6.O128,J6.O110,J6.O122,J6.O130,J6.O136,J6.O142,J6.O56,J6.O154,J6.O164,J7.O86,J7.O90,
J7.O96,J7.O101,J7.O107,J7.O112,J7.O114,J7.O119,J7.O127,J7.O106,J7.O118,J7.O126,J7.O134,
J7.O140,J7.O145,J7.O151,J7.O156,J7.O161,J7.O166,J7.O168,J7.O150,J7.O160,J8.O87,J8.O93,
J8.O99,J8.O103,J8.O111,J8.O123,J8.O131,J8.O137,J8.O143,J8.O147,J8.O155,J8.O165,J9.O8,
J9.O12,J9.O20,J9.O24,J9.O36,J9.O43,J9.O47,J9.O51,J9.O53,J9.O57,J9.O66,J10.O88,J10.O94,
J10.O21,J10.O25,J10.O29,J10.O33,J10.O38,J10.O44,J10.O104,J10.O116,J10.O124,J10.O132,
J10.O138,J10.O144,J10.O148,J10.O158,J11.O89,J11.O95,J11.O100,J11.O105,J11.O117,J11.O125,
J11.O133,J11.O139,J11.O54,J11.O58,J11.O63,J11.O67,J11.O71,J11.O75,J11.O149,J11.O159,
J12.O90,J12.O96,J12.O101,J12.O107,J12.O112,J12.O114,J12.O119,J12.O127,J12.O106,
J12.O118,J12.O126,J12.O134,J12.O140,J12.O145,J12.O151,J12.O156,J12.O161,J12.O166,J12.O168,
J12.O150,J12.O160,J13.O11,J13.O13,J13.O22,J13.O26,J13.O30,J13.O34,J13.O40,J13.O45,
J13.O27,J13.O42,J13.O46,J13.O50,J13.O52,J13.O55,J13.O59,J13.O64,J13.O68,J13.O72,J13.O76,
J13.O61,J13.O69,J14.O91,J14.O97,J14.O23,J14.O109,J14.O121,J14.O129,J14.O135,J14.O141,
J14.O146,J14.O152,J14.O157,J14.O162,J14.O167,J14.O169,J14.O153,J14.O163,J15.O92,
J15.O98,J15.O102,J15.O108,J15.O113,J15.O115,J15.O120,J15.O128,J15.O110,J15.O122,J15.O130,
J15.O136,J15.O142,J15.O56,J15.O154,J15.O164,J16.O93,J16.O99,J16.O103,J16.O111,J16.O123,
J16.O131,J16.O137,J16.O143,J16.O147,J16.O155,J16.O165,J17.O36,J17.O43,J17.O47,J18.O116,
J18.O124,J18.O132,J19.O117,J19.O125,J19.O133,J20.O118,J20.O126,J20.O134,J21.O42,
J21.O46,J21.O50,J22.O121,J22.O129,J22.O135,J23.O122,J23.O130,J23.O136,J24.O123,J24.O131,
J24.O137  /;

SET JOBPRIM(JOB,OPER) CONJUNTO DE PRIMEIRA OPERACAO DO JOB
 / J1.O4,J2.O81,J3.O82,J4.O83,J5.O84,J6.O85,J7.O86,J8.O87,J9.O8,J10.O88,J11.O89,
J12.O90,J13.O11,J14.O91,J15.O92,J16.O93,J17.O36,J18.O116,J19.O117,J20.O118,
J21.O42,J22.O121,J23.O122,J24.O123  /;

SET JOBULT(OPER) CONJUNTO DE PRIMEIRA OPERACAO DO JOB
 / O66,O69,O158,O159,O163,O164,O160,O165,O47,O132,O133,O134,O50,O135,O136,O137  /;

SET RECJOB(REC,JOB) CONJUNTO DE JOBS DO PROCESSO
/R1.J1,R1.J2,R1.J3,R1.J4,R1.J5,R1.J6,R1.J7,R1.J8,R2.J9,R2.J10,R2.J11,R2.J12,R3.J13,
R3.J14,R3.J15,R3.J16,R4.J17,R4.J18,R4.J19,R4.J20,R5.J21,R5.J22,R5.J23,R5.J24 /;

SET RECREC(REC,RECC) CONJUNTO DE RECURSOS SINCRONOS
 / R1.R1,R2.R2,R3.R3, R4.R4, R5.R5  /;

SET JOBJOB(JOB,JOBB) CONJUNTO DE RECURSOS SINCRONOS
 / J1.J1,J2.J2,J3.J3,J4.J4,J5.J5,J6.J6,J7.J7,J8.J8,J9.J9,J10.J10,J11.J11,J12.J12,
J13.J13,J14.J14,J15.J15,J16.J16,J17.J17,J18.J18,J19.J19,J20.J20,J21.J21,J22.J22,
```

```
J23.J23,J24.J24,J1.J9,J3.J10,J4.J11,J7.J12,J2.J13,J5.J14,J6.J15,J8.J16,J1.J17,
J3.J18,J4.J19,J7.J20,J2.J21,J5.J22,J6.J23,J8.J24,J9.J17,J10.J18,J11.J19,J12.J20,
J13.J21,J14.J22,J15.J23,J16.J24  /;

PARAMETER DADOS(OPER) TEMPOS DE DURACAO DAS OPERACOES
 / O4 52,O81 52,O82 52,O83 52,O84 52,O85 52,O86 52,O87 52,O8 21,O88 21,O89 21
O90 21,O11 23,O91 23,O92 23,O93 23,O12 33,O94 33,O95 33,O96 33,O13 34,O97 34
O98 34,O99 34,O20 35,O100 35,O21 46,O101 46,O22 26,O102 26,O23 58,O103 58,O24 51
O104 51,O105 51,O106 51,O25 47,O107 47,O26 35,O108 35,O27 80,O109 80,O110 80,O111 80
O29 58,O112 58,O30 76,O113 76,O33 0,O114 0,O34 0,O115 0,O36 21,O116 21,O117 21
O118 21,O38 18,O119 18,O40 98,O120 98,O42 45,O121 45,O122 45,O123 45,O43 23,O124 23
O125 23,O126 23,O44 100,O127 100,O45 118,O128 118,O46 89,O129 89,O130 89,O131 89,O47 43
O132 43,O133 43,O134 43,O50 23,O135 23,O136 23,O137 23,O51 98,O138 98,O139 98,O140 98,
O52 67,O141 67,O142 67,O143 67,O53 45,O144 45,O54 65,O145 65,O55 43,O146 43,O56 77
O147 77,O57 32,O148 32,O149 32,O150 32,O58 83,O151 83,O59 32,O152 32,O61 43,O153 43
O154 43,O155 43,O63 76,O156 76,O64 54,O157 54,O66 117,O158 117,O159 117,O160 117,O67 0
O161 0,O68 0,O162 0,O69 27,O163 27,O164 27,O165 27,O71 96,O166 96,O72 10,O167 10,O75 79
O168 79,O76 59,O169 59 /;

PARAMETER MAQUINA(MAQ)  QUANTIDADE MAQUINAS IDENTICAS
 / M1 1,M2 1,M3 1,M4 1,M5 1,M6 1,M7 1,M8 19,M9 1,M10 1,M11 1 /;

PARAMETER ALTURA(JOB)  ALT DE RECORRENCIA DE CADA PROCESSO
 / J1 10,J2 10 ,J3 10 ,J4 10 ,J5 10 ,J6 10 ,J7 10 ,J8 10 ,J9 5 ,J10 5 ,J11 5 ,J12 5
J13 5 ,J14 5 ,J15 5 ,J16 5 ,J17 1 ,J18 1 ,J19 1 ,J20 1 ,J21 1 ,J22 1 ,J23 1 ,J24 1 /;

PARAMETER RECURSO(REC)  REC DISP PARA DISTRIBUIR ENTRE OS PROCS PARALELOS
 / R1 80,R2 20,R3 20,R4 4,R5 4 /;

VARIABLES       TAO                   FLUXO DO SISTEMA
                W                     FLUXO OTIMO
                PER                   PERIODO OTIMO
                TI(OPER)              TEMPO INICIAL DE CADA OPERACAO
                O(JOB,OPER)           TEMPO EXECUCAO DE CADA OPERACAO
                K(OPER,OPERR,MAQ)     PARAMETRO DE MAQUINA
                K1(OPER,OPERS)        PARAMETRO DE SEQUENCIA DE OPER NA MAQ
                X(OPER,MAQ,MAQQP)     PARAMETRO DE MAQUINA
                U(OPER)               TEMPO INICIAL MOD PERIODO
                C(JOB)                PARAMETRO DO PROCESSO
POSITIVE VARIABLES TAO,U,TI,PER;
INTEGER VARIABLES D;
BINARY VARIABLES K,C,X,K1;
SCALAR  B                     DICOTOMIA;
B = 2356;
EQUATIONS
PREC1(JOB,OPER,OPERR)      PRECEDENCIA ENTRE OPERACOES DO  MESMO JOB
PREC1A(REC,RECC)           OPERACOES INICIAIS DE RECURSOS DIFERENTES COM MESMA PRECEDENCIA
PREC1B(REC,RECC)           DEVEM COMECAR AO MESMO TEMPO
```

```
PREC1C(OPER,OPERR,OPERS)   OPERACOES COM MESMA PRECEDENCIA COMECA AO MESMO TEMPO
PREC1D(JOB,JOBB,OPER,OPERR)
PREC2(REC)            DETERMINA QUE PELO MENOS UM JOB SEJA EXECUTADO
PREC3(REC)            LIMITA A QUANTIDADE DE JOBS IDENTICOS QUE PODEM SER EXECUTADOS
LIMINF(JOB,OPER)      LIMITA INFERIORMENTE A VARIAVEL O
LIMSUP(JOB,OPER)      LIMITA SUPERIORMENTE A VARIAVEL O
LIM(OPER)             IMPOE LIMITES A VARIAVEIS (U=<1-TAO)
TTINICIAL1(JOB)       ESTABELECE A ALTURA DE RECORRENCIA
TTINICIAL2(JOB)       ESTABELECE A ALTURA DE RECORRENCIA
TTINICIAL3(JOB)       ASSEGURA QUE A OPER NAO SEJA REALIZADA SE O SEU JOB NAO FOR REALIZADO
KDEC3(OPER,MAQ)            NUMERO BINARIO QUE INDICA QUAL OPERACAO EXECUTAR
MAQUB1(OPER,OPERR,MAQ,JOB) IMPOSSIBILITA SIMULTANEIDADE DE OPER NUMA MAQ COM OPER UNICAS
MAQUB2(OPER,OPERR,OPERS,OPES,MAQ,MAQQP,JOB,JOBB)
KDECB1(OPER,OPERR,OPERS,OPES,MAQ)
KDECB2(OPER,OPERR,OPERS,OPES,MAQ)
KDECB3(OPER,OPERR,OPERS,OPES,MAQ)
RELACIONA(JOB,JOBB)        RELACIONA JOBS QUE CONTENHAM AS MESMAS OPERACOES
OBJETIVO                   FUNCAO OBJETIVO;


RELACIONA(JOB,JOBB)$(JOBJOB(JOB,JOBB))..C(JOB)=E=C(JOBB);

PREC2(REC)..SUM((JOB)$(RECJOB(REC,JOB)), C(JOB)) =G=1;

PREC3(REC)..SUM((JOB)$(RECJOB(REC,JOB)), C(JOB)) =L=RECURSO(REC);

LIMINF(JOB,OPER)$(HABJOB(JOB,OPER))..D(JOB,OPER)=G=0;

LIMSUP(JOB,OPER)$(HABJOB(JOB,OPER))..D(JOB,OPER)=L=2;

TTINICIAL1(JOB)..SUM((OPER)$(HABJOB(JOB,OPER)), D(JOB,OPER)) =L=ALTURA(JOB);

TTINICIAL2(JOB)..SUM((OPER)$(HABJOB(JOB,OPER)), D(JOB,OPER)) =G=C(JOB);

TTINICIAL3(JOB)..SUM((OPER)$(HABJOB(JOB,OPER)), U(OPER))=L=B*C(JOB);

KDEC3(OPER,MAQ)$(HABMAQ(OPER,MAQ))..SUM((MAQQP)$(MAQP(MAQ,MAQQP)), X(OPER,MAQ,MAQQP)) =E=  1;

LIM(OPER)$(OPERACOES(OPER))..U(OPER)=L=1-TAO;

PREC1(JOB,OPER,OPERR)$(HABJOB(JOB,OPER) AND HABJOB(JOB,OPERR) AND HABOPE(OPER,OPERR))..
TAO*DADOS(OPER) =L= U(OPERR)-U(OPER)+ D(JOB,OPER) + 1 - C(JOB);

PREC1A(REC,RECC)$(RECREC(REC,RECC))..SUM((OPER,JOB)$(JOBPRIM(JOB,OPER)AND
RECJOB(REC,JOB)), U(OPER))=E=SUM((OPER,JOBB)$(JOBPRIM(JOBB,OPER)AND RECJOB(RECC,JOBB)), U(OPER));

PREC1B(REC,RECC)$(RECREC(REC,RECC))..SUM((JOB)$(RECJOB(REC,JOB)),
C(JOB))=E=SUM((JOB)$(RECJOB(RECC,JOB)), C(JOB));
```

```
PREC1C(OPER,OPERR,OPERS)$((ORD(OPER) LT ORD(OPERR))AND HABOPE(OPERS,OPER)AND
HABOPE(OPERS,OPERR) AND NOT(JOBULT(OPERS))).U(OPERR)=E=U(OPER);

PREC1D(JOB,JOBB,OPER,OPERR)$(HABJOB(JOB,OPERR) AND (HABJOB(JOBB,OPER))AND
HABOPE(OPER,OPERR)AND (NOT(JOBJOB(JOB,JOBB)))AND (NOT(JOBJOB(JOBB,JOB)))).
TAO*DADOS(OPER) =L= U(OPERR)-U(OPER)+ D(JOB,OPER)+ 1 - C(JOB);

MAQUB1(OPER,OPERR,MAQ,JOB)$(HABOPESEQ(OPER,OPERR,MAQ)AND HABJOB(JOB,OPER)).
TAO*DADOS(OPERR)+U(OPERR)=L=U(OPER)+K(OPER,OPERR,MAQ) + 1 - C(JOB);

MAQUB2(OPER,OPERR,OPERS,OPES,MAQ,MAQQP,JOB,JOBB)$(HABOPESEQ(OPER,OPERR,MAQ)AND
HABOPESEQ(OPERS,OPES,MAQ)AND MAQP(MAQ,MAQQP)AND(ORD(OPER) NE ORD(OPERS))
AND HABJOB(JOB,OPER) AND HABJOB(JOBB,OPERS)AND HABJOB(JOB,OPERS) AND HABJOB(JOBB,OPES)).
TAO*DADOS(OPERR)+U(OPERR)=L=U(OPERS)+K(OPERS,OPERR,MAQ) + 4 - X(OPER,MAQ,MAQQP) -
X(OPERS,MAQ,MAQQP) - C(JOB) -C(JOBB);

KDECB1(OPER,OPERR,OPERS,OPES,MAQ)$(HABOPESEQ(OPER,OPERR,MAQ)AND
HABOPESEQ(OPERS,OPES,MAQ)AND(ORD(OPER) LT ORD(OPERS))).
K(OPER,OPERR,MAQ)+K(OPERS,OPES,MAQ)=G= 1;

KDECB2(OPER,OPERR,OPERS,OPES,MAQ)$(HABOPESEQ(OPER,OPERR,MAQ)AND
HABOPESEQ(OPERS,OPES,MAQ)AND(ORD(OPER) LT ORD(OPERS))).
K(OPER,OPES,MAQ)+K(OPERS,OPERR,MAQ)=L= 1;

KDECB3(OPER,OPERR,OPERS,OPES,MAQ)$(HABOPESEQ(OPER,OPERR,MAQ)AND
HABOPESEQ(OPERS,OPES,MAQ)AND(ORD(OPER) LT ORD(OPERS))).
K(OPER,OPERR,MAQ)+K(OPERS,OPES,MAQ)-K(OPER,OPES,MAQ)-K(OPERS,OPERR,MAQ)=E= 1;

OBJETIVO.. W=E=TAO;

MODEL  CICLICO /ALL/;
CICLICO.WORKSPACE=50;
SOLVE     CICLICO USING MIP MAXIMIZING W;

PARAMETER TINICIAL(JOB,OPER,MAQ)     CALCULA O TEMPO INICIAL DE CADA OPERACAO;
TI.L(OPER)=(U.L(OPER)/TAO.L)$(TAO.L GT 0);
PARAMETER PERIODO     CALCULA A DURACAO DO PERIODO;
PER.L=(1/TAO.L)$(TAO.L GT 0);
DISPLAY TI.L,PER.L,C.L,TAO.L;
```

### C.3.1 Resoluçao

A seguir será apresentada a solução deste problema gerada pelo GAMS, porém como a saída do GAMS inclui a formulação apresentada anteriormente, apenas a parte relativa a solução será apresentada:

```
PJSC                                    05/17/06 15:08:16  PAGE     73
Model Statistics    SOLVE CICLICO USING MIP FROM LINE 1440  GAMS Rev 116  Windows NT/95/98

MODEL STATISTICS

BLOCKS OF EQUATIONS      20     SINGLE EQUATIONS     4999
BLOCKS OF VARIABLES       7     SINGLE VARIABLES     2786
NON ZERO ELEMENTS     17762     DISCRETE VARIABLES   2648

GENERATION TIME      =        0.391 SECONDS    2.4 Mb      WIN194-116
EXECUTION TIME       =        0.391 SECONDS    2.4 Mb      WIN194-116
PJSC                                          05/17/06 15:08:16  PAGE     74
                                GAMS Rev 116  Windows NT/95/98

               S O L V E      S U M M A R Y
     MODEL   CICLICO             OBJECTIVE  W
     TYPE    MIP                 DIRECTION  MAXIMIZE
     SOLVER  OSL                 FROM LINE  1440

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE                0.0064
 RESOURCE USAGE, LIMIT          4.051   300000.000
 ITERATION COUNT, LIMIT      1921     15000000
 OSL Version 1 Aug  7, 2000 WIN.OS.OS 19.4 058.040.038.WAT
 Work space requested by user   --   30.00 Mb
 Work space requested by solver --    5.07 Mb
**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
PJSC                                            05/17/06 15:08:16  PAGE     75
E x e c u t i o n                          GAMS Rev 116  Windows NT/95/98

----   1494 VARIABLE  TI.L  TEMPO INICIAL DE CADA OPERACAO
O4   156.000,  O8   135.000, O20   44.000, O23   68.000,  O24   79.000
O36  156.000,  O43   20.000, O47   43.000, O51   86.000,  O53  156.000
O57  124.000,  O66  156.000, O91   11.000, O97   34.000,  O109 156.000
O121  99.000,  O129 144.000, O135  76.000, O141  99.000,  O146   9.000
O152  52.000,  O153  87.000, O157  84.000, O162 156.000,  O163 130.000
O167 156.000,    O169   9.000

----   1494 VARIABLE  PER.L                =      157.000  PERIODO OTIMO
----   1494 VARIABLE  C.L  PARAMETRO DO PROCESSO
J1  1.000,  J5  1.000,  J9  1.000,  J14 1.000,  J17 1.000,  J22 1.000
----   1494 VARIABLE  TAO.L                =        0.006  FLUXO DO SISTEMA
**** REPORT FILE SUMMARY
TESEA C:\USUARIOS\DENISE\ALGOTESE\TESEA.PUT
TESEB C:\USUARIOS\DENISE\ALGOTESE\TESEB.PUT
```

```
TESEC C:\USUARIOS\DENISE\ALGOTESE\TESEC.PUT
TEMPOS C:\USUARIOS\DENISE\ALGOTESE\TEMPOS.PUT
ALTREL C:\USUARIOS\DENISE\ALGOTESE\ALTREL.PUT

EXECUTION TIME       =        0.032 SECONDS    1.9 Mb      WIN194-116
PJSC                               05/17/06 15:08:16  PAGE     76
                                   GAMS Rev 116  Windows NT/95/98
USER: Leo Pini Magalhaes                            G000925:1529AP-WIN
      UNICAMP, FEE/DCA                                          DC2914
**** FILE SUMMARY
INPUT      C:\USUARIOS\DENISE\ALGOTESE\MODELO3.GMS
OUTPUT     C:\USUARIOS\DENISE\ALGOTESE\MODELO3.LST
```

# Índice