ISSN -

MODELOS DE OTIMIZAÇÃO DE REDES DE FILAS ABERTAS PARA PROJETO E PLANEJAMENTO DE *JOB-SHOPS*

Gabriel R. Bitran
M.I.T.

Reinaldo Morábito
UFSCar

Resumo

Neste artigo examinamos modelos de otimização baseados na teoria de redes de filas abertas para o projeto e planejamento de sistemas de manufatura discretos, como *job-shops*. Duas categorias de problemas são consideradas: A primeira minimiza o investimento de capital de maneira a atingir uma medida de desempenho (estoque em processo, *leadtime* de produtos), a segunda tenta otimizar a medida de desempenho sujeito às restrições de disponibilidade de recursos. Diversas abordagens de solução conhecidas da literatura são apresentadas e analisadas. A metodologia é ilustrada com um exemplo de uma rede *job-shop* derivado de uma aplicação numa indústria de semicondutores.

Palavras-chave: rede de filas aberta; otimização e avaliação de desempenho; projeto e planejamento de *job-shop;* sistemas de manufatura discretos.

Abstract

In this paper we review optimization models based on open queueing networks for the design and planning of discrete manufacturing systems, such as job-shops. We consider two categories of problems: The first minimizes capital investment subject to attaining a performance measure (work-in-process, product leadtime), the second seeks to optimize the performance measure subject to resource constraints. Different solution approaches known in the literature are presented and analyzed. The methodology is illustrated with a job-shop example derived from an actual application in the semiconductor industry.

Keywords: open queueing networks; optimization and performance evaluation; job-shop design and planning; discrete manufacturing systems.

1. Introdução

Job-shops são complexos sistemas discretos de manufatura que processam uma grande variedade de produtos ou *jobs* em pequenos lotes. Tipicamente os *jobs* percorrem fluxos complexos ao longo das estações de trabalho (*shops*) e formam filas de espera defronte as máquinas. Neste artigo analisamos alguns modelos de otimização, baseados na teoria de *redes de filas abertas* (*open queueing networks*), para o projeto e planejamento de sistemas *job-shops*. Exemplos das decisões envolvidas são: seleção de produtos e tecnologia, escolha de equipamentos e capacidade, e alocação de produtos a plantas.

Para o propósito do artigo, agrupamos os problemas em apenas duas classes propostas em Bitran e Dasu (1992): *desempenho desejado do sistema* (SP1 - *Strategic problem* 1) e *desempenho ótimo do sistema* (SP2). Uma discussão de outras classes (p.e., *partição da instalação* - SP3) pode ser encontrada em Bitran e Dasu (1992) e Bitran e Morábito (1994).

Na classe SP1 o objetivo é minimizar o investimento no sistema sujeito às restrições dos desempenhos desejados do sistema. Típicas medidas de desempenho são: *estoque em processo* (*work in process - WIP*), *leadtime* de produtos (tempo total para fabricar ou montar o produto), taxa de produção, utilização de equipamentos. Nos problemas a seguir escolhemos o WIP como medida de desempenho; outras medidas poderiam ter sido escolhidas. Considere o seguinte exemplo da classe SP1:

(SP1.1) *WIP desejado:*

- *Objetivo:* minimizar o custo de aquisição de equipamentos
- *Variáveis de decisão:* capacidade de cada estação, tecnologia
- *Restrições:* limitante superior para o nível de WIP.

Na classe SP2, desejamos otimizar o desempenho do sistema sujeito às limitações de capital para investimento no sistema. Um exemplo da classe SP2 é dado abaixo:

(SP2.1) *WIP ótimo:*

- *Objetivo:* minimizar o nível de WIP
- *Variáveis de decisão:* capacidade de cada estação, tecnologia
- *Restrições:* limitante superior para o custo de aquisição de equipamentos.

Note que SP1.1 e SP2.1 envolvem um *tradeoff* entre o capital de investimento e o capital de trabalho. Neste artigo revemos modelos de otimização e abordagens de solução para problemas do tipo SP1.1 e SP2.1, utilizando teoria de rede de filas. Alguns dos algoritmos aqui analisados (algoritmos 1, 2 e 3 da seção 5.1) são refinamentos daqueles examinados em Bitran e Morábito (1994, 1995b). Na seção 2, discutimos brevemente como representar o sistema *job-shop* por meio de uma rede de filas e avaliar suas medidas de desempenho. Na seção 3, definimos modelos de otimização genéricos para SP1.1 e SP2.1 e nas seções 4 e 5, estes modelos são analisamos para as redes de Jackson e redes de Jackson generalizadas, respectivamente. Para ilustrar, apresentamos alguns resultados computacionais de um exemplo de rede *job-shop* de uma fábrica de semicondutores. Finalmente, na seção 6 apresentamos as conclusões do artigo.

2. Avaliação de medidas de desempenho do sistema *job-shop*

O sistema *job-shop* pode ser representado como uma rede de filas *aberta*, onde os nós correspondem às estações de trabalho (*shops*), e os arcos ligando os nós, aos fluxos de *jobs* entre as estações. A rede de filas é aberta porque os *jobs* chegam na rede, recebem processamento em uma ou mais estações, e eventualmente saem da rede. Se os processos de chegada e processamento (serviço) de *jobs* nas estações forem assumidos estocásticos, filas de *jobs* são esperadas defronte às estações.

Em geral os modelos consideram que o sistema esteja em estado de *equilíbrio* (*steady-state*). Os *jobs* podem chegar nas estações individualmente ou em lotes. Eles podem pertencer a uma ou múltiplas classes (famílias) de produtos, e percorrer roteiros determinísticos ou probabilísticos ao longo das estações. Esses roteiros ainda podem ser cíclicos ou acíclicos. Cada estação pode ter uma ou várias máquinas, que processam *jobs* individualmente ou em lotes. As filas de espera podem ter capacidade limitada ou ilimitada, com diversas disciplinas de atendimento possíveis, por exemplo, *first-come first-served* (FCFS), *shortest-processing-time-first* (SPT).

Se os processos de chegada e serviço em cada estação forem Markovianos, então a rede de filas é chamada *rede de Jackson* (i.e., rede de filas M/M/m); caso contrário, *rede de Jackson generalizada* (rede de filas GI/G/m). Redes de Jackson possuem soluções exatas em forma de produto que simplificam bastante sua análise. Por outro lado, redes de Jackson generalizadas são bem mais difíceis de serem analisadas, o que têm motivado diversos autores para o desenvolvimento de métodos aproximados para avaliar suas medidas de desempenho. Um exemplo é o chamado *método de decomposição* (Whitt, 1983).

Por ilustração, a seguir apresentamos resumidamente como utilizar o método de decomposição para avaliar o nível de WIP de redes de Jackson generalizadas com múltiplas classes, roteiros determinísticos (cíclicos ou acíclicos), uma máquina em cada estação, filas de espera ilimitadas, e disciplina FCFS. Assumimos que os *jobs* chegam e são processados individualmente nas estações. Este método também pode ser aplicado em redes com roteiros probabilísticos, múltiplas máquinas em cada estação, chegada e processamento em lote, combinação e criação de *jobs* nas estações, e outras disciplinas além de FCFS. Para maiores detalhes, veja por exemplo Segal e Whitt (1989), Bitran e Tirupati (1991), Kouvelis e Tirupati (1991), e Buzacott e Shanthikumar (1993).

2.1 Método de decomposição

O método de decomposição envolve basicamente três passos: no passo 1 a rede é decomposta num conjunto de estações independentes, no passo 2 as medidas de desempenho (no caso, o WIP) são avaliadas para cada estação, analisada como um sistema de filas individual, e no passo 3 as medidas de desempenho são avaliadas para a rede, combinando os resultados do passo 2.

O método assume que tanto os intervalos de tempo entre chegadas quanto os tempos de processamento dos *jobs* nas estações são variáveis aleatórias *independentes e identicamente distribuídas*; além disso, suas distribuições são aproximadas por apenas dois parâmetros: a média e o *coeficiente quadrático de variação* (*squared coefficient of variation* - scv), que é um parâmetro de variabilidade definido pela razão entre a variância e o quadrado da média. Considere a seguinte notação para os dados de entrada:

r	número de classes de produtos na rede
n	número de estações da rede
n_k	número de operações do roteiro da classe k (k=1,..,r)
λ'_k	taxa média de chegada externa de produtos da classe k
ca'_k	scv do intervalo de tempo entre chegadas externas de produtos da classe k
n_{kl}	estação que processa a l-ésima operação do roteiro da classe k (l=1,..,n_k)
m_j	número de máquinas na estação j (no caso, m_j=1, j=1,..,n)
μ_j	taxa média de processamento de cada máquina da estação j
cs_j	scv do tempo de processamento da estação j
a_j, b_j	coeficientes de custo de capacidade da estação j
v_j	valor monetário médio de um *job* na estação j, independente de sua classe.

As tabelas 1 e 2 apresentam estes parâmetros para o exemplo de uma rede *job-shop* com r=10 classes de produtos e n=13 estações. Este é um exemplo real de uma fábrica de semicondutores, analisada em Bitran e Tirupati (1989a) e Bitran e Morábito (1995c). A figura l ilustra o roteiro de cada classe (coluna n_{kl} da tabela l) ao longo das estações. Note que para produzir uma unidade do produto da classe 1, um *job* entra na rede e visita primeiro a estação 1, em seguida as estações 2, 4, 2, 9 e 10, e finalmente a estação 11, para depois sair da rede (este roteiro está indicado na figura Ia). Em cada uma destas n_1=7 visitas (i.e. roteiro da classe 1), o *job* recebe uma operação diferente. A última linha da tabela l apresenta a taxa média de produção da rede, igual a 10 unidades de produtos por unidade de tempo.

Por simplicidade, consideramos cada estação j como uma única máquina (i.e., m_j=1), com *capacidade* ou taxa média de processamento μ_j (a extensão para o caso com $m_j \geq 1$ é simples). Também consideramos que essa taxa média de processamento independe da classe do produto e da operação. Caso contrário, as taxas médias de processamento para cada classe k e cada operação l na estação j podem ser combinadas para aproximar a *taxa agregada média* μ_j, conforme seção 5.3 em Bitran e Morábito (1995a). Convém observar que, em redes onde *jobs* de uma mesma classe k visitam várias vezes uma estação j, se em cada visita eles recebem uma operação l diferente (i.e. as taxas médias μ_{kl} na estação j variam com a operação l), então a rede sob cenas condições pode resultar instável, mesmo que os processos de chegada e processamento sejam Markovianos, a disciplina seja FCFS, e as utilizações médias sejam menores do que l (veja a discussão em Bramson (1994) sobre uma rede de classe única com roteiro l, 2, 2, ..., 2, l).

Tabela 1: Dados de entrada das classes de produtos da rede *job-shop*

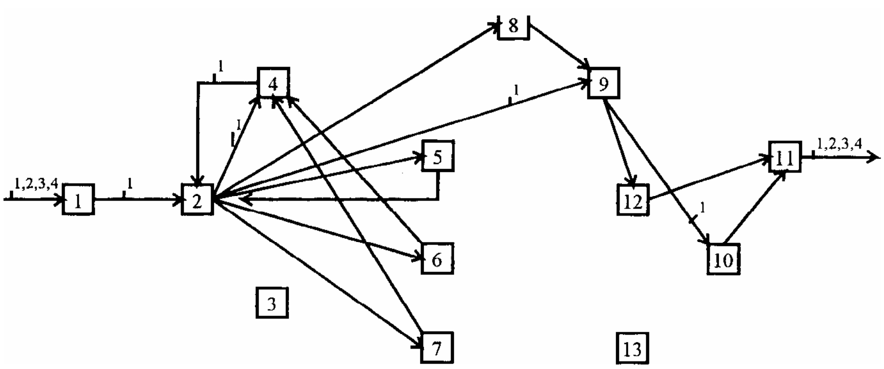
Classe k	λ'_k	**ca'$_k$**	**n_{kl}**	**n_k**
1	1.0	0.500	1, 2, 4, 2, 9, 10, 11	7
2	1.0	0.500	1, 2, 5, 2, 8, 9, 10, 11	8
3	1.0	0.333	1, 2, 6, 4, 2, 9, 12, 11	8
4	1.0	0.333	1, 2, 7, 4, 2, 9, 10, 11	8
5	1.0	0.333	1, 2, 4, 12, 2, 9, 2, 13	8
6	1.0	0.333	1, 2, 5, 12, 2, 9, 7, 13	8
7	1.0	0.250	1, 2, 6, 12, 2, 8, 2, 13	8
8	1.0	1.000	1, 2, 3, 7, 4, 12, 2, 8, 6, 9, 2, 13	12
9	1.0	1.000	1, 2, 3, 5, 4, 6, 12, 2, 8, 2, 10, 6, 13	13
10	1.0	0.333	1, 2, 3, 6, 2, 4, 12, 7, 2, 9, 11, 5, 13	13
Total	**10.0**			**93**

Tabela 2: Dados de entrada das estações da rede *job-shop*

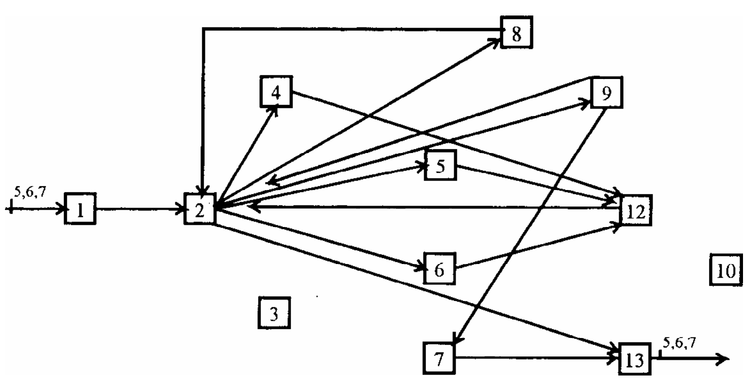
Estação j	μ_j	cs_j	v_j	a_j	b_j
1	13.004	0.500	100	5.68	-51.69
2	27.778	0.250	1612	2.59	-50.40
3	3.160	0.333	733	74.77	-165.40
4	10.000	0.500	1052	6.93	-48.53
5	5.631	0.333	912	12.62	-49.73
6	9.225	0.250	1683	7.51	-48.54
7	5.999	1.000	1662	11.11	-46.67
8	4.500	0.333	1812	27.66	-87.11
9	10.000	0.333	1730	7.47	-52.27
10	5.711	0.333	1600	15.34	-61.30
11	5.441	0.333	1882	27.03	-102.94
12	7.440	0.500	1486	13.01	-67.74
13	7.502	0.500	3250	14.22	-74.67
Total	**115.391**				

Figura 1: Roteiros de cada uma das 10 classes ao longo das 13 estações da *redejob-shop.*

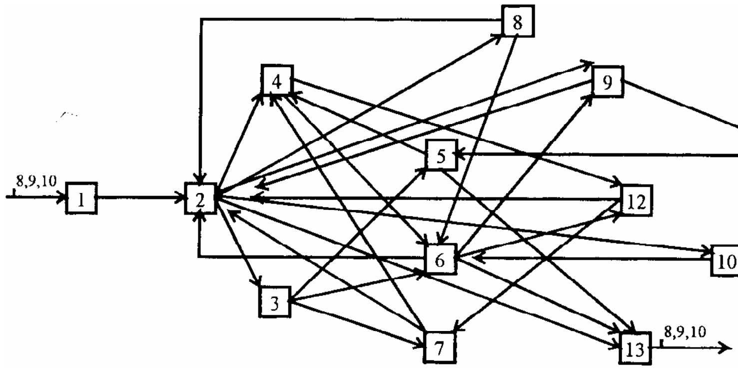
(a) Classes 1, 2, 3 e 4



(b) Classes 5, 6 e 7



(c) Classes 8, 9 e 10



No passo 1, a rede de filas original é decomposta num conjunto de filas "independentes", correspondendo às estações. O objetivo é rescrever os parâmetros iniciais $\{\lambda'_k, ca'_k, n_{kl}, k=1,..,r; l=1,..,n_k; \mu_j, cs_j, j=1,..,n\}$ como $\{\lambda_j, ca_j, \mu_j, cs_j, j=1,..,n\}$, onde λ_j e ca_j agora denotam, respectivamente, a taxa média de chegada e o scv do intervalo de tempo entre chegadas de produtos na estação j. Note que desta forma estamos agregando todos os produtos em uma única classe, denominada *classe agregada*, e analisando seu fluxo ao longo da rede.

Cada parâmetro λ_j é obtido somando-se as taxas médias de chegada externa de todas as classes que visitam a estação j, ponderadas pelo número de visitas definido nos seus roteiros (p.e., note na tabela 1 que o roteiro da classe 10 faz três visitas na estação 2), ou seja:

$$\lambda_j = \Sigma_{k=1,..,r}\, \lambda'_k\, \Sigma_{l=1,..,n_k}\, \delta(n_{kl}=j)$$

onde $\delta(n_{kl}=j)=1$, se $n_{kl}=j$, e $\delta(n_{kl}=j)=0$, caso contrário.

Os parâmetros ca_j são aproximadamente estimados pela solução de um sistema linear nas variáveis ca_j, cd_j e $cd_{k,l}$, definido por (Bitran e Tirupati, 1988):

$$ca_j = \Sigma_{k=1,..,r}\, (\lambda'_k/\lambda_j)\, \Sigma_{l=1,..,nk}\, cd_{k,l-1}\, \delta(n_{kl}=j) \quad \text{para } j=1,..,n$$

$$cd_j = (\lambda_j/\mu_j)^2\, cs_j + [1 - (\lambda_j/\mu_j)^2]\, ca_j \quad \text{para } j=1,..,n$$

$$cd_{k,l} = (\lambda'_k/\lambda_j)\, cd_{nkl} + [1 - (\lambda'_k/\lambda_j)]\, (\lambda'_k/\lambda_j) + [1 - (\lambda'_k/\lambda_j)]^2\, cd_{k,l-1}$$

$$\text{para } k=1,..,r;\ l=1,..,n_k.$$

onde $cd_{k,0}=ca'_k$, e cd_j (ou cd_{nkl} com $n_{kl}=j$) e $cd_{k,l}$ são variáveis implícitas. A tabela 3 apresenta os parâmetros λ_j e ca_j obtidos para o exemplo das tabelas 1 e 2 (as demais colunas da tabela 3 estão definidas a seguir).

Tabela 3: Parâmetros e medidas de desempenho obtidos para o exemplo das tabelas 1 e 2

Estação j	λ_j	ca_j	ρ_j	L_j	v_jL_j	F_j
1	10.0	0.492	0.769	1.974	197.377	288.325
2	25.0	0.601	0.900	4.298	6929.058	598.457
3	3.0	0.760	0.949	10.694	7838.522	223.823
4	7.0	0.608	0.700	1.569	1650.825	207.700
5	4.0	0.613	0.710	1.500	1367.930	120.093
6	6.0	0.583	0.650	1.118	1881.392	191.332
7	4.0	0.619	0.667	1.715	2850.717	119.836
8	4.0	0.665	0.889	4.403	7979.090	168.193
9	8.0	0.642	0.800	2.327	4025.622	224.300
10	4.0	0.662	0.700	1.489	2382.308	150.240
11	5.0	0.684	0.919	6.194	11656.346	240.055
12	7.0	0.614	0.941	9.226	13709.098	216.225
13	6.0	0.677	0.800	2.653	8620.970	240.110
Total				**49.160**	**71089.253**	**2988.689**

No passo 2, medidas de desempenho da classe agregada em cada estação podem ser estimadas, tais como utilização da capacidade, número médio de *jobs* na estação, WIP. Por exemplo, a utilização média na estação j, definida por $\rho_j=\lambda_j/\mu_j$, pode ser facilmente calculada com λ_j e μ_j. Também podemos calcular o WIP (em valores monetários) na estação j, definido por $v_jL_j(\lambda_j,ca_j,\mu_j,cs_j)$, onde $L_j(\lambda_j,ca_j,\mu_j,cs_j)$ é o número médio de *jobs* em fila e em serviço na estação j, como uma função das médias e dos scv do tempo entre chegadas e do tempo de processamento da classe agregada na estação j. Este número pode ser aproximadamente calculado por (Whitt, 1983):

$$L_j(\lambda_j,ca_j,\mu_j,cs_j) = \{(\lambda_j/\mu_j)^2\ (ca_j+cs_j)\ g(\lambda_j,ca_j,\mu_j,cs_j)\ /\ 2[1-(\lambda_j/\mu_j)]\} + (\lambda_j/\mu_j) \quad (1)$$

com:

$$g(\lambda_j,ca_j,\mu_j,cs_j) = \begin{cases} \exp\{-2[1-(\lambda_j/\mu_j)](1-ca_j)^2\ /\ 3(\lambda_j/\mu_j)(ca_j+cs_j)\}, & \text{se } ca_j<1, \\ 1, & \text{se } ca_j\geq 1 \end{cases}$$

Também podemos definir o custo de alocar a capacidade μ_j na estação j (investimento em capacidade) pela função quadrática:

$$F_j(\mu_j)= a_j\ \mu_j^2+ b_j\ \mu_j + c_j \quad (2)$$

onde a_j, b_j e c_j são coeficientes conhecidos. Por conveniência, assumimos que é possível adicionar capacidade na estação j em quantidades pequenas o suficiente para considerar a capacidade $_j$ *contínua*.

A tabela 3 apresenta os valores da utilização média (ρ_j), do número médio de *jobs* (L_j), do WIP (v_jL_j), e do custo de capacidade (F_j) de cada estação j da rede *job-shop* das tabelas 1 e 2 (por simplicidade, assumimos c_j=0).

Finalmente, no passo 3, medidas de desempenho são avaliadas para a rede como um todo, recompondo-se os resultados do passo 2. Por exemplo, somando-se o WIP de todas as estações, obtemos o WIP da rede $\sum_{j=1,..,n} v_j L_j$, com valor 71089 (tabela 3). Note que este valor refere-se a classe agregada; ao desagregarmos estes resultados, obtemos o WIP para cada classe original.

Para maiores detalhes do método de decomposição, veja por exemplo Whitt (1983), Kouvelis e Tirupati (1991), Suri et al (1993) e Bitran e Morábito (1994, 1995a).

3. Modelos de Otimização para SP1.1 e SP2.1

Nesta seção apresentamos dois modelos de otimização genéricos para os problemas SP1.1 e SP2.1 da seção 1. Considere a seguinte notação (assumimos que as classes de produtos são agregadas conforme método de decomposição discutido na seção 2):

$F_j(\mu_j, m_j)$ custo (ou investimento) de alocar a capacidade $(_j, m_j)$ na estação j
F_T orçamento disponível para a capacidade da rede
$L_j(\mu_1, m_1;_2, m_2; ...; \mu_n, m_n)$ o número médio de *jobs* na estação j, como uma função da capacidade da rede
W_T limitante superior para o WIP da rede.

O problema de WIP desejado SP1.1 é o problema de determinar a capacidade (μ_j, m_j) para cada estação j de maneira a minimizar o custo e satisfazer a restrição do nível desejado de WIP na rede, W_T. SP1.1 é formulado como:

$$
\begin{array}{lll}
\text{(SP1.1)} & \text{minimizar} & \Sigma_{j=1,\dots,n} F_j(\mu_j, m_j) \\
 & \text{sujeito a:} & \Sigma_{j=1,\dots,n} v_j L_j(\mu_1, m_1; {}_2, m_2; ...; \mu_n, m_n) \le W_T \\
 & \text{com:} & (\mu_j, m_j) \in P_j, \quad j=1,..,n
\end{array}
$$

onde P_j é um dado domínio das variáveis, e $L_j(\mu_1, m_1; {}_2, m_2; ...; \mu_n, m_n)$ deriva da teoria de rede de filas (compare com a expressão (1)).

Similarmente, o problema de WIP ótimo SP2.1 é o problema de determinar a capacidade (μ_j, m_j) para cada estação j de maneira a minimizar o WIP da rede e satisfazer a restrição do orçamento disponível, F_T. SP2.1 é formulado como:

$$
\begin{array}{lll}
\text{(SP2.1)} & \text{minimizar} & \Sigma_{j=1,..,n} v_j L_j(\mu_1, m_1; {}_2, m_2; ...; \mu_n, m_n) \\
 & \text{sujeito a:} & \Sigma_{j=1,..,n} F_j(\mu_j, m_j) = F_T \\
 & \text{com:} & (\mu_j, m_j) \in P_j, \quad j=1,..,n.
\end{array}
$$

onde $F_j(\mu_j, m_j)$ é uma função conhecida, conforme a expressão (2).

Diversos autores apresentaram abordagens para os dois modelos acima; a seguir revemos algumas delas. Com a finalidade de apresentá-las de uma forma mais estruturada, adotaremos a notação sugerida em Bitran e Dasu (1992) que denota cada instância do problema por $\alpha/\beta/\chi/\delta$, onde $\alpha \in$ {SP1.1, SP2.1}, $\beta \in \{J, G\}$, $\chi \in \{S, M\}$ e $\delta \in \{R, N\}$. O símbolo α indica o tipo de problema, β indica se o problema é aplicado a uma rede de Jackson (J) ou a uma rede de Jackson generalizada (G), χ indica se as estações têm uma única máquina (S) ou podem ter múltiplas máquinas (M), e δ indica a variável de decisão do modelo: taxa média de processamento (R) ou número de máquinas (N) em cada estação.

Por conveniência, nos modelos ./././R a seguir, assumimos que toda capacidade na rede seja *homogênea* e *intercambiável* entre as estações. Um exemplo é a mão-de-obra treinada que pode ser transferida de uma estação para outra. Os algoritmos apresentados a seguir também podem ser aplicados quando a capacidade de uma estação não é transferível para todas as outras (Bitran e Morábito, 1995c).

4. Redes de Jackson (Modelos ./J/./.)

Nas redes de Jackson, ao contrário das redes de Jackson generalizadas, pode ser mostrado que cada estação j pode ser tratada como se fosse um sistema de fila "estocasticamente independente" (Buzacott e Shanthikumar, 1993; Suri et al., 1993). Dessa maneira, $L_j(\mu_1, m_1; \mu_2, m_2; ...;\mu_n, m_n)$ em SP1.1 e SP2.1 reduz-se a uma função apenas de μ_j e m_j, ao invés de uma função de μ_1, m_1; μ_2, m_2; ...; μ_n, m_n.

4.1 Modelos ./J/./R

Kleinrock (1964, 1976) estudou o problema de minimizar o número médio de *jobs* em redes de Jackson com uma máquina em cada estação (i.e. m_j=1). As variáveis de decisão são as taxas de serviço μ_j, j=1,..,n. Kleinrock mostrou que $L_j(\mu_j)=\lambda_j/(\mu_j-\lambda_j)$ e assumiu que o custo F_j seja proporcional a μ_j (i.e. $F_j(\mu_j)=f_j\mu_j$, onde f_j é o custo unitário da capacidade na estação j). Ele propôs o seguinte modelo para SP2.1/J/S/R (compare com SP2.1 da seção 3):

$$
\begin{array}{lll}
\text{(SP2.1/J/S/R)} & \text{minimizar} & \Sigma_{j=1,..,n} L_j(\mu_i) \\
 & \text{sujeito a:} & \Sigma_{j=1,..,n} f_j\mu_j = F_T \\
 & \text{com:} & \mu_j \geq 0, \quad j=1,..,n
\end{array}
$$

Introduzindo o multiplicador de Lagrange y associado à restrição de igualdade em SP2.1/J/S/R, a função lagrangeana resulta em:

$$\Sigma_{j=1,..,n} L_j(\mu_i) + y\,(\Sigma_{j=1,..,n} f_j\, \mu_j - F_T).$$

Ao derivar esta função em relação à cada $_j$, a solução ótima de SP2.1/J/S/R é obtida em forma fechada, dada por:

$$\mu_j^* = \lambda_j + [(f_j\lambda_j) / \Sigma_{i=1,..,n}\sqrt{(f_i\lambda_i)}]\, [(F_T - \Sigma_{i=1,..,n} f_i\lambda_i) / f_j].$$

Cinco condições são satisfeitas no modelo SP2.1/J/S/R (Bitran e Dasu, 1992):

(i) $L_j(\mu_j)$ é uma função convexa de μ_j (o número médio de *jobs* na estação j é uma função convexa da capacidade da estação j),
(ii) $L_j(\mu_j)$ não depende de μ_i, $i \neq j$, $i=1,..,n$ (adições de capacidade em outras estações têm nenhum efeito no número médio de *jobs* da estação j),
(iii) μ_j é contínuo (as variáveis de decisão são variáveis contínuas),
(iv) $F_j(\mu_j)$ é uma função convexa de μ_j (o custo de capacidade da estação j é uma função convexa da capacidade da estação j),
(v) $L_j(\mu_j)$ pode ser expresso em forma fechada.

Estas condições serão úteis para analisar os demais modelos deste artigo. As condições (i)-(iv) reduzem o modelo num programa convexo, que pode ser resolvido otimamente via métodos exatos de ótimo-local (p.e., Bazaraa et al., 1993). A condição (v) permite que a solução do problema seja em forma fechada.

Podemos formular SP2.1/J/M/R exatamente como SP.2.1/J/S/R; Harel e Zipkin (1987) mostraram que $L_j({}_j)$ também é uma função convexa de ${}_j$ para $m_j>1$. Assim, as condições (i)-(iv) são satisfeitas e SP2.1/J/M/R também é um programa convexo. SP1.1/J/S/R e SP1.1/J/M/R podem ser definidos e analisados de maneira similar.

4.2 Modelos ./J/./N

As variáveis de decisão agora são inteiras e correspondem ao número de máquinas m_j, $j=1,..,n$, em cada estação. Assim como nos modelos ./J/./R (seção 4.1), também pode ser mostrado nos modelos ./J/./N que $L_j(m_j)$ é uma função convexa e decrescente em m_j, independente de m_i, $i \neq j$, $i=1,..,n$ (condições (i) e (ii) satisfeitas).

Em SP1.1/J/M/N, desejamos encontrar a solução de custo mínimo satisfazendo um nível de WIP menor ou igual a um limite especificado W_T (veja SP1.1 na seção 3). Boxma et al. (1990) assumiram que $F_j(m_j)$, o custo de alocar m_j máquinas na estação j, seja uma função convexa não-decrescente de m_j (condição (iv) satisfeita), e modelaram este problema de WIP desejado (também chamado *problema de alocação de servidores*) como:

$$\begin{array}{lll} \text{(SP1.1/J/M/N)} & \text{minimizar} & \Sigma_{j=1,..,n} F_j(m_j) \\ & \text{sujeito a:} & \Sigma_{j=1,..,n} v_j L_j(m_j) \leq W_T \\ & \text{com:} & m_j \geq \lceil \lambda_j/\mu_j \rceil, \quad m_j \text{ inteiro, } j=1,..,n. \end{array}$$

(onde $\lceil z \rceil$ denota o menor inteiro maior ou igual a z). Note que SP1.1/J/M/N é um programa convexo, porém com variáveis inteiras (condição (iii) violada). Boxma et al. propuseram um simples algoritmo iterativo guloso (heurístico) para resolver SP1.1/J/M/N, e apresentaram limitantes para o erro da solução (veja também a abordagem em Sundarraj et al (1994) para um problema similar). O algoritmo começa com a menor alocação possível de máquinas para cada estação. Em cada iteração, ele adiciona uma máquina na estação com mínimo *índice de prioridade*, definido por meio de *análise marginal* como o quociente entre o aumento de custo e a redução de WIP na estação. O algoritmo termina assim que a adição de uma máquina resultar numa alocação viável. Experimentos computacionais resultaram em erros relativos em torno de 5%.

Mais recentemente, Frenk et al. (1994) apresentaram 2 algoritmos para resolver SP1.1/J/M/N, que podem ser vistos como extensões do algoritmo de Boxma et al. (1990). Estes algoritmos produzem soluções melhores do que o de Boxma et al., porém, com um esforço computacional maior (eles têm complexidade $O(Mn^2)$ e $O(Mn^3)$, respectivamente, enquanto o algoritmo de Boxma et al tem complexidade O(Mn), onde M é o máximo número de máquinas entre as chamadas *alocações não-dominadas*).

Em SP2.1/J/M/N, desejamos alocar (ou realocar) máquinas de maneira a otimizar a medida de desempenho, ou seja, minimizar o WIP na rede (veja SP2.1 na seção 3). Boxma et al. (1990) assumiram que um total de M máquinas homogêneas deve ser alocado às estações, e modelaram este problema de WIP ótimo (também chamado de *problema de realocação de servidores*) como:

(SP2.1/J/M/N)	minimizar	$\Sigma_{j=1,..,n}\, v_j\, L_j(m_j)$
	sujeito a:	$\Sigma_{j=1,..,n}\, m_j = M$
	com:	$m_j \geq \lceil \lambda_j/\mu_j \rceil$, m_j inteiro, j=1,..,n.

Novamente, temos um programa convexo (condições (i), (ii) e (iv) satisfeitas), porém com variáveis inteiras (condição (iii) violada). Boxma et al apresentaram um algoritmo iterativo guloso para resolver SP2.1/J/M/N, com complexidade O(Mn). Apesar deste algoritmo ser similar ao algoritmo heurístico apresentado para SP1.1/J/M/N, ele é exato (i.e. tem garantia de otimalidade). O algoritmo começa com a menor alocação de máquinas possível para cada estação. Em cada iteração, ele adiciona uma máquina na estação com o máximo índice de prioridade, definido como a redução de WIP por unidade de máquina na estação. O algoritmo termina quando todas as M máquinas tiverem sido alocadas.

Para maiores detalhes destes e outros modelos e algoritmos para as redes de Jackson, veja por exemplo Bitran e Morábito (1995b) e Buzacott e Shanthikumar (1993).

5. Redes de Jackson Generalizadas (Modelos ./G/./.)

Nas redes de Jackson generalizadas, ao contrário das redes de Jackson, o número médio de *jobs* na estação j, L_j, em SP1.1 e SP2.1 depende de μ_1, m_1; μ_2, m_2; ...; μ_n, m_n.

5.1 Modelos ./G/./R

Nos modelos ./G/./R as variáveis de decisão correspondem à taxa de serviço $_j$, j=1,..,n, em cada estação (condição (iii) satisfeita). Seja $F_j(\mu_j)$ o custo de alocar capacidade μ_j na estação j, definida como uma função convexa não-decrescente e diferenciável de μ_j (condição (iv) satisfeita).

Bitran e Tirupati (1989a) abordaram SP1.1/G/S/R (também chamado *problema de redistribuição ótima de WIP*) e SP2.1/G/S/R (*problema de redistribuição ótima de recursos*). Eles assumiram que: (a) o scv cs_j seja independente à pequenas mudanças na capacidade μ_j (i.e. a variância e o quadrado da média do tempo de processamento na estação j variam na mesma proporção e assim, cs_j se mantém aproximadamente constante), e (b) L_j depende apenas de μ_j, ao invés de $\mu_1, \mu_2, ..., \mu_n$ (condição (ii) satisfeita). A hipótese (b) parece se verificar a medida que o número de classes aumenta e a proporção da demanda devida por cada classe diminui (veja discussão em Bitran e Tirupati (1988), Whitt (1988), Bitran e Morábito (1995a)). Sob tal hipótese, eles mostraram que a função $L_j(\mu_j)$ é convexa em μ_j (condição (i) satisfeita), e modelaram SP1.1/G/S/R e SP2.1/G/S/R como programas convexos, dados por:

(SP1.1/G/S/R)	minimizar	$\Sigma_{j=1,..,n} F_j(\mu_j)$
	sujeito a:	$\Sigma_{j=1,..,n} v_j L_j(\mu_j) \leq W_T$
	com:	$\lambda_j > \mu_j$, j=1,..,n.
(SP2.1/G/S/R)	minimizar	$\Sigma_{j=1,..,n} v_j L_j(\mu_j)$
	sujeito a:	$\Sigma_{j=1,..,n} F(\mu_j) = F_T$
	com:	$\mu_j > \lambda_j$, j=1,..,n.

Experimentos computacionais com redes de r=10 classes de produtos (roteiros determinísticos) e com n=13 estações resultaram em erros nos valores dos scv ca_j, j=1,...,n, menores que 3% (sob hipótese (b)), o que sugere que os ca_j são pouco sensíveis à mudanças na capacidade. Para uma descrição detalhada dos algoritmos utilizados, veja Bitran e Tirupati (1989a) e Bitran e Morábito (1995b) (veja também Wein (1990) para uma abordagem baseada no *movimento Browniano* para o problema SP2.1/G/S/R com apenas uma classe de produtos e roteiro probabilístico).

Uma abordagem mais refinada do que em Bitran e Tirupati (1989a) foi apresentada em Bitran e Sarkar (1994), que atualiza os scv ca_j, j=1,..,n, ao longo das iterações e desta maneira, reflete a dependência de cada L_j com relação a $\mu_1, \mu_2, ..., \mu_n$. Considere inicialmente o problema de WIP desejado; ele pode ser resolvido pelo seguinte algoritmo iterativo exato:

Algoritmo 1:

Passo 0: Dados os parâmetros iniciais $\{\lambda'_k, ca'_k, n_{kl}, k=1,..,r; l=1,..,n_k, \mu^0_j, cs_j, j=1,..,n\}$, aplique o método de decomposição (seção 2) para obter os parâmetros $\{\lambda_j, ca^0_j, \mu^0_j, cs_j, j=1,..,n\}$, onde ca^0_j e μ^0_j denotam, respectivamente, o scv inicial do intervalo de tempo entre chegadas e a capacidade inicial na estação j. Defina $W_T = \Sigma_{j=1,..,n} v_j L_j(\mu^0_j)$ e faça p=1.

Passo 1: Em cada iteração p, utilize os scv ca^{p-1}_j, j=1,..,n, para resolver o seguinte programa convexo nas variáveis μ_j:

$$\min \Sigma_{j=1,..,n} F_j(\mu_j)$$

$$\text{s.a.: } \Sigma_{j=1,..,n} v_j L_j(\mu_j) = W_T$$

$$\text{com: } \mu_j > \lambda_j, j=1,..,n.$$

onde $L_j(\mu_j)$ e $F_j(\mu_j)$ são definidos conforme (1) e (2). Sejam $\mu^p{}_j$, j=1,..,n, denotando a solução ótima do problema acima usando $ca^{p-1}{}_j$.

Passo 2: Aplique o método de decomposição com os parâmetros {λ'_k, ca'_k, n_{kl}, k=1,..,r; l=1,..,n_k, $\mu^p{}_j$, cs_j, j=1,..,n} para obter os parâmetros {λ_j, $ca^p{}_j$, $\mu^p{}_j$, cs_j, j=1,..,n}. Pare se $ca^{p-1}{}_j$ e $ca^p{}_j$ forem suficientemente próximos; caso contrário, faça p=p+1 e volte para o passo 1.

Note que, em cada iteração, o algoritmo 1 assume que o scv ca_j seja independente a mudanças na capacidade das estações (veja passo 1). Dado que a função $L_j(\mu_j)$ é convexa em μ_j (Bitran e Tirupati, 1989a), resulta que o problema do passo 1 é convexo nas em variáveis μ_j, j=1,..,n, podendo ser resolvido através das diversas técnicas de programação convexa (veja, p.e., Bazaraa et al., 1993). Bitran e Sarkar (1994) mostraram que o algoritmo converge para uma solução ótima sob condições usualmente encontradas na prática.

Ao aplicarmos o algoritmo 1 na rede *job-shop* da seção 2, obtemos uma solução com custo de capacidade 2278 para o nível de WIP 71089 (veja tabela 4). O algoritmo converge após duas iterações, para uma precisão de 0.001 nos valores dos ca_j. Note que os valores dos ca_j obtidos são muito próximos dos valores originais (compare as tabelas 3 e 4), apesar da variação da capacidade μ_j nas estações.

Tabela 4: Parâmetros e medidas de desempenho referentes ao algoritmo 1

Estação j	ca_j	μ_j	ρ_j	L_j	v_jL_j	F_j
1	0.492	10.390	0.962	13.114	1311.434	76.112
2	0.598	26.978	0.927	5.841	9415.943	525.354
3	0.760	3.275	0.916	6.355	4658.410	260.353
4	0.607	8.143	0.860	3.730	3923.519	64.327
5	0.616	4.720	0.847	3.039	2772.018	46.459
6	0.581	7.215	0.832	2.489	4189.145	40.719
7	0.617	5.255	0.761	2.686	4463.872	61.536
8	0.657	4.660	0.858	3.402	6163.572	194.739
9	0.638	9.270	0.863	3.465	5994.499	157.403
10	0.657	4.868	0.822	2.664	4262.773	65.111
11	0.672	5.690	0.879	4.047	7616.165	289.389
12	0.604	7.923	0.884	4.537	6741.802	279.960
13	0.668	7.330	0.819	2.946	9576.101	216.651
total		**105.717**		**58.315**	**71089.253**	**2278.113**

A solução produzida pelo algoritmo 1 nos indica que podemos reduzir substancialmente os recursos (de 2989 para 2278), sem aumentar o WIP (71089). Esta redução é obtida redistribuindo-se eficientemente este WIP entre as estações, por meio da variação da capacidade μ_j de cada estação (compare as tabelas 3 e 4). Note que a redistribuição de WIP não implica numa mudança de processo ou tecnologia, nem numa alteração da taxa de produção original da rede (igual a 10 produtos por unidade de tempo).

Parte do WIP das estações 3, 8, 11 e 12 é "transferido" para as demais estações (tabelas 3 e 4); os recursos aumentam um pouco nestas estações, porém são reduzidos em mais de 50% nas estações 1, 4, 6 e 10. Se todos os valores v_j, j=1,.,n, forem iguais, o algoritmo 1 redistribui o número médio de produtos (ao invés do WIP) da rede ao longo das estações, de maneira a minimizar os recursos na rede. Além disto, se a_j=0, c_j=0 e b_j=1, j=1,..,n, então $F_j(\mu_j)=\mu_j$, e o algoritmo 1 agora determina a mínima capacidade necessária para manter o número médio de produtos W_T na rede.

Um algoritmo iterativo exato similar ao algoritmo 1 também pode ser utilizado para resolver o problema de WIP ótimo; as mesmas suposições com respeito a convexidade de L_j e a convergência do algoritmo 1 são assumidas no algoritmo abaixo:

Algoritmo 2:

Passo 0: Dados os parâmetros iniciais $\{\lambda'_k, ca'_k, n_{kl}, k=1,..,r; l=1,..,n_k, \mu^0_j, cs_j, j=1,..,n\}$, aplique o método de decomposição (seção 2) para produzir os parâmetros $\{\lambda_j, ca^0_j, \mu^0_j, cs_j, j=1,..,n\}$, onde ca^0_j e μ^0_j denotam, respectivamente, o scv inicial do intervalo de tempo entre chegadas e a capacidade inicial na estação j. Defina $F_T = \sum_{j=1,..,n} F_j(\mu^0_j)$ e faça p=1.

Passo 1: Em cada iteração p, utilize os scv ca^{p-1}_j, j=1,..,n, para resolver o seguinte programa convexo nas variáveis μ_j:

$$\min \sum_{j=1,..,n} v_j\, L_j(\mu_j)$$

$$\text{s.a.: } \sum_{j=1,..,n} F_j(\mu_j) = F_T$$

$$\text{com: } \mu_j > \lambda_j,\ j=1,..,n.$$

onde $L_j(\mu_j)$ e $F_j(\mu_j)$ são definidos conforme (1) e (2). Sejam μ^p_j, j=1,..,n, denotando a solução ótima do problema acima usando ca^{p-1}_j.

Passo 2: Aplique o método de decomposição com os parâmetros $\{\lambda'_k, ca'_k, n_{kl}, k=1,..,r; l=1,..,n_k, \mu^p_j, cs_j, j=1,..,n\}$ para obter os parâmetros $\{\lambda_j, ca^p_j, \mu^p_j, cs_j, j=1,..,n\}$. Pare se ca^{p-1}_j e ca^p_j forem suficientemente próximos; caso contrário, faça p=p+1 e volte para o passo 1.

Ao aplicarmos o algoritmo 2 na rede *job-shop* da seção 2, obtemos uma solução com nível de WIP 49254, para o custo de capacidade 2989 (veja tabela 5). O algoritmo converge após duas iterações, para uma precisão de 0.001 nos valores dos ca_j. Note que estes valores são muito próximos dos valores originais (compare as tabelas 3 e 4), apesar da variação da capacidade μ_j nas estações.

A solução produzida pelo algoritmo 2 nos indica que podemos reduzir substancialmente o WIP (de 71089 para 49254), sem alterar os recursos (2989). Esta redução é obtida redistribuindo-se eficientemente os recursos entre as estações, por meio da variação da capacidade μ_j de cada estação (compare as tabelas 3 e 5). A redistribuição dos recursos não implica numa mudança de processo ou tecnologia, nem numa alteração da taxa de produção original da rede.

Note que o algoritmo "vende" capacidade das estações 1, 4, 5, 6, 7, 9 e 10 para "comprar" capacidade para as estações 2, 3, 8, 11, 12 e 13. Embora a capacidade da rede tenha se alterado (de 115.4 para 112.2), seus custos são exatamente os mesmos (2989). Se todos os valores v_j, j=1,.,n, forem iguais, o algoritmo 2 redistribui os recursos disponíveis F_T de maneira a minimizar o número médio de *jobs* na rede. Além disto, se a_j=0, c_j=0 e b_j=1, j=1,..,n, então $F_j(\mu_j)=\mu_j$, e o algoritmo 2 agora redistribui capacidade, ao invés de recursos de capacidade. Neste caso, dizemos que estamos *balanceando* o sistema (lembre-se que estamos assumindo que toda capacidade é homogênea e intercambiável).

Tabela 5: Parâmetros e medidas de desempenho referentes ao algoritmo 2

Estação j	ca_j	μ_j	ρ_j	L_j	v_jL_j	F_j
1	0.492	10.604	0.943	8.609	860.861	90.541
2	0.602	28.041	0.892	3.966	6392.554	623.270
3	0.761	3.421	0.877	4.279	3136.301	309.222
4	0.610	8.712	0.804	2.587	2721.365	103.173
5	0.621	5.081	0.787	2.141	1952.440	73.096
6	0.589	7.818	0.767	1.787	3007.797	79.567
7	0.624	5.828	0.686	1.874	3115.223	105.356
8	0.665	4.999	0.800	2.369	4292.979	255.741
9	0.643	9.918	0.807	2.415	4178.749	216.376
10	0.666	5.296	0.755	1.891	3026.341	105.638
11	0.682	6.050	0.826	2.795	5260.484	366.620
12	0.611	8.403	0.833	3.101	4607.640	349.405
13	0.678	7.987	0.751	2.062	6701.743	310.684
Total		**112.158**		**39.876**	**49254.477**	**2988.689**

5.1.1 Modelos ./G/./R com alternativas discretas para mudança de capacidade

Bitran e Tirupati (1989b) também analisaram SP1.1/G/M/R com *alternativas discretas* para mudança de capacidade em cada estação. Ao invés de escolher (μ_j, m_j), j=1,..,n, como variáveis de decisão, eles consideraram um número finito de alternativas para mudança de capacidade em cada estação. Considere a seguinte notação para os dados de entrada:

n_j número total de alternativas para a estação j, j=1,..,n
m_{ji} número de máquinas (idênticas) da estação j na alternativa i, i=1,.., n_j
μ_{ji} taxa média de processamento de cada máquina da estação j na alternativa i
F_{ji} custo (ou investimento) de capacidade da estação j na alternativa i.

Defina y_{ji} como uma variável de decisão 0-1 na estação j, satisfazendo $\sum_{i=1,.,nj} y_{ji}=1$, tal que:

y_{ji} = 1, se a alternativa i for escolhida na estação j,
0, caso contrário

A escolha y_{ji}=1 implica em alocar a capacidade ($_{ji}$, m_{ji}) na estação j, a um custo F_{ji}. Por conveniência e sem perda de generalidade, considere que todas as alternativas em todas as estações envolvem apenas uma máquina (i.e. m_{ji}=1, j=1,..,n; i=1,..,n_j). A tabela 6 ilustra 5 possíveis alternativas de capacidade para cada uma das n=13 estações da rede *job-shop* das tabelas 1 e 2 (i.e. μ_{ji}; j=1,..,13; i=1,..,5); note que a primeira alternativa (coluna 1) corresponde a capacidade original da tabela 2. Os recursos F_{ji} podem ser obtidos substituindo cada μ_{ji} na expressão (2).

Tabela 6: Alternativas discretas para mudança de capacidade nas estações

Estação j	**1**	**2**	**3**	**4**	**5**
1	13.004	10.5	11.0	14.0	15.0
2	27.778	26.0	27.0	28.0	30.0
3	3.160	3.5	3.5	4.0	4.5
4	10.000	7.5	8.0	9.0	11.0
5	5.631	4.5	4.7	5.0	6.0
6	9.225	6.5	7.0	9.0	12.0
7	5.999	5.0	5.5	6.0	6.5
8	4.500	4.5	5.0	5.5	6.0
9	10.000	8.5	9.0	10.0	11.0
10	5.711	4.5	4.7	5.0	6.0
11	5.441	5.3	5.6	5.9	6.0
12	7.440	7.5	8.0	8.5	9.0
13	7.502	6.5	7.0	7.5	8.0
Total	**115.391**				

Bitran e Tirupati (1989b) assumiram que o número médio de *jobs* na estação j na alternativa i, L_{ji}, dependa apenas da capacidade na estação j, ao invés da capacidade de todas as estações. Desta maneira, $L_{ji}=L_j(\lambda_j, ca_j, \mu_{ji}, cs_j)$ pode ser calculado por meio da expressão (1) para cada estação j e cada alternativa i. SP1.1/G/S/R com alternativas discretas de capacidade pode ser formulado pelo seguinte programa linear inteiro (o modelo para SP1.1/G/M/R é similar):

(SP1.1/G/S/R) minimizar $\Sigma_{j=1,..,n}\ \Sigma_{i=1,..,nj}\ F_{ji}\ y_{ji}$

sujeito a: $\Sigma_{j=1,..,n}\ \Sigma_{i=1,..,nj}\ v_j\ L_{ji}\ y_{ji} \leq W_T$

$\Sigma_{i=1,..,nj}\ y_{ji} = 1,\ j=1,..,n$

com: $y_{ji} \in \{0,1\},\ \ j=1,..,n,\ i=1,..,n_j$

Bitran e Tirupati (1989b) mostraram que: (i) a solução ótima da relaxação LP de SP1.1/G/S/R (ou SP1.1/G/M/R) tem *zero* ou *duas* diferentes variáveis y_{ji} com valores fracionários e, (ii) se esta solução ótima tiver *duas* variáveis com valores fracionários, então elas correspondem à mesma estação. Baseados nestes resultados, eles propuseram um algoritmo heurístico para resolver o problema acima (para detalhes do algoritmo, veja Bitran e Tirupati (1989b) e Bitran e Morábito (1995b)).

Um refinamento desta abordagem é atualizar os scv ca_j, j=1,..,n, ao longo das iterações, conforme algoritmos 1 e 2. A seguir adaptamos o algoritmo 1 para SP1.1/G/S/R com alternativas discretas (o algoritmo 2 também pode ser adaptado para SP2.1/G/S/R de maneira similar):

Algoritmo 3 (Algoritmo 1 para alternativas discretas)

Passo 0: Dados os parâmetros iniciais $\{\lambda'_k, ca'_k, n_{kl}, k=1,..,r; l=1,..,n_k, \mu^0_j, cs_j, j=1,..,n\}$, aplique o método de decomposição (seção 2) para obter os parâmetros $\{\lambda_j, ca^0_j, \mu^0_j, cs_j, j=1,..,n\}$, onde ca^0_j e μ^0_j denotam respectivamente o scv inicial do intervalo entre chegadas, e a capacidade inicial da estação j (p.e., $\mu^0_j=\mu_{j1}$, j=1,..,n). Defina W_T e faça p=1.

Passo 1: Em cada iteração p, utilize os scv ca^{p-1}_j, j=1,..,n, para computar L_{ji} em (1) para cada μ_{ji}, e resolva o seguinte problema de programação linear inteira nas variáveis y_{ji}:

$$\min \sum_{j=1,..,n} \sum_{i=1,..,nj} F_{ji}\, y_{ji}$$

$$\text{s.t.: } \sum_{j=1,..,n} \sum_{i=1,..,nj} v_j\, L_{ji}\, y_{ji} \leq W_T$$

$$\sum_{i=1,..,nj} y_{ji} = 1, \;\; j=1,..,n$$

$$\text{with: } y_{ji} \in \{0,1\}, \;\; j=1,..,n;\; i=1,..,n_j.$$

Seja y^p_{ji}, j=1,..,n, i=1,..,n_j, denotando a solução ótima do problema acima usando ca^{p-1}_j. Note que se y^p_{ji}=1, então a capacidade μ_{ji} é alocada na estação j. Seja μ^p_j, j=1,..,n, a capacidade alocada na estação j.

Passo 2: Aplique o método de decomposição com os parâmetros $\{\lambda'_k$, ca'_k, n_{kl}, k=1,..,r; l=1,..,n_k, μ^p_j, cs_j, j=1,..,n$\}$ para obter os parâmetros $\{\lambda_j$, ca^p_j, μ^p_j, cs_j, j=1,..,n$\}$. Pare se ca^{p-1}_j e ca^p_j forem suficientemente próximos ou se p atingir um certo limite de iterações; caso contrário, faça p=p+1 e volte para o passo 1.

O problema do passo 1 pode ser resolvido pelas técnicas de programação linear inteira conhecidas da literatura (veja p.e. Nemhauser e Wolsey, 1988), ou pelo procedimento heurístico proposto em Bitran e Tirupati (1989b) mencionado acima, que demanda pouco esforço computacional para encontrar soluções relativamente boas. No presente artigo utilizamos um procedimento exato do tipo *branch-and-bound* para resolvê-lo.

Ao aplicarmos o algoritmo 3 na rede *job-shop* (com as 5 alternativas da tabela 6), obtemos a solução da tabela 7 com custo de capacidade 2359 e nível de WIP 70928. O algoritmo converge após três iterações para uma precisão de 0.001 nos valores dos ca_j e um desvio relativo máximo de 0.2% do valor da solução ótima. Note que ele escolheu diferentes alternativas para as estações da rede (veja segunda coluna da tabela 7).

Tabela 7: Parâmetros e medidas de desempenho referentes ao algoritmo 3

Est. j	Alt. i	ca_{ji}	μ_{ji}	ρ_{ji}	L_{ji}	v_jL_{ji}	F_{ji}
1	2	0.492	10.500	0.952	10.315	1031.498	83.475
2	3	0.598	27.000	0.926	5.782	9321.135	527.310
3	2	0.760	3.500	0.857	3.652	2676.847	337.032
4	3	0.610	8.000	0.875	4.229	4448.631	55.280
5	4	0.619	5.000	0.800	2.285	2084.087	66.850
6	3	0.584	7.000	0.857	2.954	4971.318	28.210
7	3	0.622	5.500	0.727	2.266	3765.365	79.392
8	1	0.660	4.500	0.889	4.386	7947.184	168.120
9	3	0.637	9.000	0.889	4.300	7438.939	134.640
10	4	0.654	5.000	0.800	2.348	3756.943	77.000
11	3	0.671	5.600	0.893	4.597	8651.750	271.197
12	3	0.606	8.000	0.875	4.216	6265.355	290.720
13	4	0.666	7.500	0.800	2.637	8568.902	239.850
total			**106.100**		**53.967**	**70927.955**	**2359.077**

Os valores dos ca_j obtidos pelo algoritmo 3 são muito próximos dos seus valores originais (compare as tabelas 3, 4 e 7). Note que o WIP obtido (70928) é menor do que W_T (71089), ao invés de exatamente igual a W_T conforme solução produzida pelo algoritmo 1 (verifique que o problema do passo 1 no algoritmo 3 tem uma desigualdade, diferente do problema do algoritmo 1). Note também que, dado que $\mu_{ji}>\lambda_j$ para todo i e j, segue que a necessidade de recursos produzida pelo algoritmo 1 (igual a 2278) resulta num limitante inferior para a necessidade de recursos produzida pelo algoritmo 3 (2359).

Nossas primeiras experiências computacionais com o algoritmo 3 sugerem que ele converge sob tolerâncias razoáveis para a precisão dos valores finais de ca_j; a prova de sua convergência é um tópico para pesquisa futura.

Os algoritmos 1, 2 e 3 apresentados nesta seção foram codificados na linguagem de modelagem *GAMS* (*General Algebraic Mathematical System*, versão 2.25) (Brooke et al., 1992). Para resolver em cada iteração os programas linear e convexo (passo 1) dos algoritmos 1 e 2 utilizamos o *solver GAMS/MINOS*. Para resolver em cada iteração os sistemas lineares (passos 0 e 2) e o programa linear inteiro (passo 1) do algoritmo 3 utilizamos o *solver GAMS/OSL*. As soluções apresentadas nas tabelas 4, 5 e 7 foram obtidas em poucos minutos (incluindo a geração de relatórios detalhados) utilizando um microcomputador PC-AT-486DX2. O desempenho computacional ainda pode ser melhorado com a implementação de rotinas matemáticas que explorem as características particulares dos programas convexo e linear inteiro envolvidos.

5.2 Modelos ./G/./N

Baseados nos resultados de Bitran e Tirupati (1989a) para SP1.1/G/S/R e SP2.1/G/S/R (seção 5.1), Van Vliet e Rinnooy Kan (1991) assumiram que L_j depende apenas de m_j, ao invés de m_1, m_2, ..., m_n (condição (ii) satisfeita). Desta maneira, pode ser mostrado que $L_j(m_j)$ é uma função convexa de m_j (condição (i) satisfeita).

Ao assumir $F_j(m_j)$ como uma função convexa de m_j (condição (iv) satisfeita), SP1.1/G/M/N é formulado como um programa convexo, porém com variáveis inteiras (condição (iii) violada), exatamente como SP1.1/J/M/N descrito na seção 4.2. Para resolvê-lo, Van Vliet e Rinnooy Kan (1991) propuseram um algoritmo iterativo guloso (heurístico), similar ao algoritmo proposto por Boxma et al. (1990) para SP1.1/J/M/N (veja seção 4.2). Van Vliet e Rinnooy Kan também forneceram limitantes para o erro da solução heurística em relação a ótima. Experimentos computacionais com uma rede de 10 classes de produtos e 13 estações resultaram em erros de 5% a 7%, o que mostra que a hipótese de convexidade parece ser razoável.

Similarmente, SP2.1/G/M/N também é formulado como um programa convexo (condições (i), (ii) e (iv) satisfeitas), porém com variáveis inteiras (condição (iii) violada), exatamente como SP2.1/J/M/N descrito na seção 4.2. Para resolvê-lo, Van Vliet e Rinnooy Kan (1991) propuseram um algoritmo iterativo guloso, similar ao algoritmo proposto por Boxma et al. (1990) para SP2.1/J/M/N (veja seção 4.2). Este algoritmo termina após O(Mn) passos com a redistribuição ótima das M máquinas ao longo das n estações (obviamente sob a hipótese de que $L_j(m_j)$ seja uma função convexa de m_j). Experimentos computacionais indicaram que a solução ótima produzida pelo algoritmo para o suposto problema convexo pode ser utilizada como uma boa aproximação para o problema original.

Similarmente à seção 5.1, abordagens mais refinadas, baseadas na atualização dos scv ca_j em cada iteração, também podem ser aqui utilizadas, conforme algoritmos 1, 2 e 3. A decisão de utilizá-las ou não deve levar em conta o *tradeoff* entre o benefício da qualidade das soluções produzidas e o esforço computacional adicional necessário para obtê-las.

6. Conclusões

Neste artigo examinamos modelos de otimização baseados na teoria de rede de filas para problemas de WIP desejado (SP1.1) e WIP ótimo (SP2.1). Nosso enfoque foi no projeto e planejamento de sistemas *job-shops*. Exemplos de decisões envolvidas são: Qual o mínimo investimento em capacidade necessário para o sistema atingir um desempenho desejado? Como melhorar o desempenho atual do sistema com os recursos disponíveis?

Algumas abordagens conhecidas da literatura para redes de Jackson e redes de Jackson generalizadas foram discutidas. Em particular, três algoritmos para resolver SP1.1 e SP2.1 foram examinados em detalhes. Para ilustrar a aplicação da metodologia, alguns resultados computacionais de um exemplo de uma rede *job-shop* com 10 classes de produtos e 13 estações foram apresentados. Estes algoritmos também podem ser utilizados para gerar curvas de *tradeoff* entre o desempenho da rede e os recursos disponíveis, conforme Bitran e Morábito (1995c, 1995d).

Agradecimentos

Os autores gostariam de agradecer os dois *referees* pelos seus úteis comentário e sugestões. Esta pesquisa contou com o apoio da FAPESP através da concessão de uma bolsa de pós-doutorado, processo #93/0891-7.

Referências bibliográficas

BAZARAA, M. S., H. D. Sherali e C. M. Shetty (1993). *Nonlinear programming: Theory and algorithms*, 2nd.ed., Wiley, NY.

BITRAN, G. R. e S. Dasu (1992). A review of open queueing network models of manufacturing systems. *Queueing Systems* 12, 95-134.

BITRAN, G. R. e R. Morábito (1994). Open queueing networks: Optimization and performance evaluation models for discrete manufacturing systems. WP#3743-94, Sloan School of Management, MIT, 45p (aceito para publicação no *Production and Operations Management*).

BITRAN, G. R. e R. Morábito (1995a). Um exame dos modelos de redes de filas abertas aplicados a sistemas de manufatura discretos---Parte I. *Gestão & Produção* 2(2), 192-219.

BITRAN, G. R. e R. Morábito (1995b). Um exame dos modelos de redes de filas abertas aplicados a sistemas de manufatura discretos---Parte II. *Gestão & Produção* 2(3), 297-320.

BITRAN, G. R. e R. Morábito (1995c). Manufacturing systems design: Tradeoff curve analysis. WP#3805-95, Sloan School of Management, MIT, 33p (submetido para o *Production and Operations Management*).

BITRAN, G. R. e R. Morábito (1995d). An overview of tradeoff curve analysis in the design of manufacturing systems. WP#3806-95, Sloan School of Management, MIT, 28p (aceito para publicação na *Gestão & Produção*).

BITRAN, G. R. e D. Sarkar (1994). Targeting problems in manufacturing queueing networks - An iterative scheme and convergence. *European Journal of Operational Research* 76, 501-510.

BITRAN, G. R. e D. Tirupati (1988). Multiproduct queueing networks with deterministic routing: Decomposition approach and the notion of interference. *Mgmt. Sci.* 34(1), 75-100.

BITRAN, G. R. e D. Tirupati (1989a). Tradeoff curves, targeting and balancing in manufacturing queueing networks. *Operations Research* 37, 547-564.

BITRAN, G. R. e D. Tirupati (1989b). Capacity planning in manufacturing networks with discrete options. *Annals of Operations Research* 17, 119-136.

BITRAN, G. R. e D. Tirupati (1991). Approximations for network of queues with overtime. *Mgmt. Sci.* 37(3), 282-300.

BOXMA, O. J., A. Rinnooy Kan e M. Van Vliet (1990). Machine allocation problems in manufacturing networks. *European Journal of Operational Research* 45, 47-54.

BRAMSON, M., “Instability ofFIFO queueing networks”. *Armais ofAppIiedProbability* 4(2), 414-431, 1994.

BROOKE, A., D. Kendrick e A. Meeraus, *Release 2.25 GAMS - A user ‘s guide.* The Scientific Press, San Francisco, 1992.

BUZACOTT, J. A. e J. G. Shanthikumar (1993). *Stochastic models of manufacturing systems*, Prentice-Hall, Englewood Cliffs, NJ.

FRENK, H.; M. Labbe, M. Van Vliet, S. Zhang (1994). Improved algorithms for machine allocation in manufacturing systems. *Operations Research* 42, 523-530.

HAREL, A. e P. Zipkin (1987). Strong convexity results for queueing systems. *Operations Research* 35, 405-418.

KLEINROCK, L. (1964). *Communication nets: stochastic message flow and delay*, Dover Publ., NY.

KLEINROCK, L. (1976). *Queueing systems*, vol 2: Computer applications, Wiley, NY.

KOUVELIS, P. e D. Tirupati (1991). Approximate performance modeling and decision making for manufacturing systems: A queueing network optimization framework. *Journal of Intelligent Manufacturing* 2, 107-134.

NEMHAUSER, G. L e L. A. Wolsey (1988). *Integer and combinatorial optimization*, Wiley, NY, 763p.

SEGAL, M. e W. Whitt (1989). A queueing network analyzer for manufacturing. *Teletraffic Science for New Cost-Effective Systems, Networks and Services, ITC-12*, M. Bonatti (ed.), Elsevier, North-Holland, Amsterdam, 1146-1152.

SUNDARRAJ, R. P., P. S. Sundararaghavan, D. R. Fox (1994). Optimal server acquisition in open queueing networks. *J. Oper. Res. Soc.* 45(5), 549-558.

SURI, R., J. L. Sanders, M. Kamath (1993). Performance evaluation of production networks. *Handbooks in OR/MS*, S. C. Graves (ed.), vol 4, Elsevier, North-Holland, Amsterdam.

VAN VLIET, M. e A. Rinnooy Kan (1991). Machine allocation algorithms for *job* shop manufacturing. *Journal of Intelligent Manufacturing* 2, 83-94.

WEIN, L. M. (1990). Capacity allocation in generalized Jackson networks. *Oper. Res. Lett.* 15, 215-242.

WHITT, W. (1983). The queueing network analyzer. *Bell Syst. Tech. J.* 62, 2779-2815.

WHITT, W. (1988). A light-traffic approximation for single-class departures from multi-class queues. *Mgmt. Sci.* 34(11), 1333-1346.

Republicado de Pesquisa Operacional, v.15, n. 1 e 2, pp.1-22, 1995