v.2, n.3, p. 297-321, dez.1995

UM EXAME DOS MODELOS DE REDES DE FILAS ABERTAS APLICADOS A SISTEMAS DE MANUFATURA DISCRETOS: PARTE II

Gabriel R. Bitran
Sloan School of Management,
Massachusetts Institute of Technology, 02139 Cambridge,
MA, EUA.

Reinaldo Morabito
Departamento de Engenharia de Produção,
Universidade Federal de São Carlos, 13565-905
São Carlos, SP (morabito@power.ufscar.br)

Resumo

Este artigo apresenta a segunda (e última) parte do nosso exame dos modelos de redes de filas abertas aplicados a sistemas de manufatura discretos. Nosso enfoque é em modelos de projeto e planejamento de job-shops. Na primeira parte (BITRAN & MORABITO, 1995b) revisamos métodos de decomposição exatos e aproximados para modelos de avaliação de desempenho em sistemas com múltiplas classes de produtos e diversas estações de trabalho. Nesta segunda parte examinamos modelos de otimização de três categorias de problemas: a primeira minimiza o investimento de capital de maneira a atingir uma medida de desempenho (estoque em processo ou leadtime), a segunda busca otimizar a medida de desempenho sujeito às restrições de recursos, e a terceira explora resultados de pesquisas recentes com a redução de complexidade mediante reprojeto da planta e da partição de produtos.

Palavras-chave: ***redes de filas abertas, otimização e avaliação de desempenho, métodos de decomposição, sistemas discretos de manufatura, projeto e planejamento de job-shops.***

1. Introdução

Grande parte dos produtos é manufaturada em sistemas *discretos*, em que os itens são processados individualmente ou em lotes. Assim, um importante problema estratégico é o *projeto* e o *planejamento* de sistemas de manufatura discretos. Exemplos de decisões envolvidas são: seleção de produtos e tecnologia, escolha de equipamentos e capacidade, e alocação de produtos a plantas. Para o

propósito deste trabalho, agrupamos os problemas de projeto em três classes, propostas em BITRAN & DASU (1992): (i) desempenho desejado do sistema (SP1 - Strategical Problem 1), (ii) desempenho ótimo do sistema (SP2), e (iii) partição da instalação (SP3).

Na classe SP1 o objetivo é minimizar o investimento no sistema de manufatura 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 gasto pelo sistema para fabricar ou montar o produto), *taxa de produção*, e *utilização* (*intensidade de tráfego*) de equipamentos. A seguir escolhemos o WIP como medida de desempenho. 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. Finalmente, na classe SP3 procuramos subdividir o sistema de manufatura em unidades menores (que podem ser vistas como plantas dentro da planta) para melhorar o desempenho global. Entretanto, esta partição pode requerer duplicação de equipamentos e recursos. Considere o seguinte exemplo da classe SP3:

(SP3.1) *Número de produtos e WIP desejados em cada planta:*

Objetivo: minimizar o custo de aquisição de equipamentos

Variáveis de decisão: número de plantas, mix de produtos em cada planta, e capacidade de cada estação

Restrições: limitante superior para o número de produtos e o WIP em cada planta.

Note que SP3.1 também envolve um *tradeoff* entre o custo de adicionar capacidade e a redução da complexidade gerencial do sistema. Ele pode ser visto como um caso especial da classe SP1. As decisões envolvidas são: número de plantas em que subdivimos o sistema original, alocação de produtos às plantas, e escolha da capacidade de cada planta. De fato, os problemas da classe SP3 são casos especias de ambas as classes SP1 e SP2; eles estão sendo aqui considerados separadamente com a finalidade de salientar sua importância no projeto de sistemas de produção (veja seção 6).

No presente artigo apresentamos a segunda e última parte do nosso exame dos modelos de *redes de filas abertas* (*open queueing networks - OQN*) aplicados a sistemas de manufatura discretos. Nossa atenção é dirigida para as decisões de médio e longo prazo, como por exemplo no projeto e planejamento de *job-shops*, e não para aspectos mais operacionais do sistema, como programação e controle da produção. No primeiro artigo (BITRAN & MORABITO, 1995b), focalizamos os chamados *modelos de avaliação de desempenho* de OQN, que nos auxiliam a avaliar o desempenho do sistema. Nesta segunda parte, focalizamos os *modelos de otimização* de OQN, que combinam técnicas de programação matemática e a teoria de OQN. Basicamente, a diferença entre um modelo de avaliação de desempenho e um modelo de otimização reside no fato de que o primeiro é *descritivo*, isto é, por meio das medidas de desempenho ele proporciona ao usuário importantes *insights* sobre a operação do sistema numa dada configuração, enquanto o segundo é *prescritivo*, isto é, ele determina a configuração *ótima* do sistema, que

otimiza certo critério e satisfaz certas restrições impostas pelo usuário.

Num exame anterior, BITRAN & DASU (1992) analisaram diversos modelos de otimização para *job-shops*. No presente artigo aquele exame é atualizado e estendido com uma preocupação mais quantitativa. Apresentamos modelos de OQN de múltiplas classes de produtos em maiores detalhes, enfatizando a importância dos efeitos de interferência entre as classes e das aproximações de tráfego-leve. Alguns algoritmos de solução, baseados em análise marginal e na heurística gulosa, são descritos em detalhes. Também incluímos desenvolvimentos recentes como a partição de produtos, e sugerimos perspectivas para futuras pesquisas.

Na próxima seção discutimos brevemente a forma pela qual a rede de manufatura pode ser representada como uma OQN (esta seção é um resumo da seção 2 do primeiro artigo). Também fazemos referências a outros exames da literatura relacionados com nossa revisão. Na seção 3 iniciamos nossa discussão dos modelos de otimização e, nas seções 4 e 5, tais modelos são analisados para as *redes de Jackson* e as *redes de Jackson generalizadas*, respectivamente. Finalmente, na seção 6 discutimos perspectivas para pesquisa futura.

Devido à frequência com que iremos nos referir a seções e expressões do nosso primeiro artigo, optamos por simplificar a referência e utilizar apenas um asterisco após o número da seção ou da expressão. Assim, por exemplo a referência "seção 3*" corresponde a: "seção 3, daquele artigo", enquanto que a referência "seção 3" corresponde a: "seção 3, deste presente artigo". O mesmo procedimento é adotado em relação às expressões.

2. Representação em Rede de Filas de um Sistema de Manufatura Discreto

Job shops são complexos sistemas de manufatura discretos que processam uma grande variedade de produtos ou *jobs*; porém, em pequenos lotes (CHASE & AQUILANO, 1992). Em geral, *job-shops* envolvem fluxos complexos de *jobs* através dos *shops* (estações) e longas filas de espera na frente das máquinas. Podemos então representar um *job-shop* como uma *rede de filas*, onde os nós correspondem às estações e os arcos ligando os nós, aos fluxos de *jobs* entre as estações.

O estudo de redes de filas começou basicamente com o trabalho de ERLANG (1917) em telefonia. Desde então, aplicações têm aparecido em diversas áreas; por exem-plo, comunicação, computação, transporte, produção, manutenção, biologia (redes neurais), saúde (modelos comportamentais), química e materiais (polimerização), entre outras; veja DISNEY & KONIG (1985). HSU et al (1993) e SURI et al (1993) nos fornecem uma descrição abrangente do uso de redes de filas para representar sistemas de manufatura. Cada nó (estação) contém os seguintes elementos:

(i) processo de chegada,
(ii) processo de serviço, e
(iii) fila de espera.

O *processo de chegada* na estação é descrito pelo intervalo de tempo entre-chegadas de *jobs*. As notações M, G e GI indicam, respectivamente, um processo markoviano, genérico, e genérico independente (processo de renovação). Podemos ter todos os *jobs* pertencendo a uma única classe ou família, ou pertencendo a múltiplas classes diferentes. A chegada de *jobs* na estação pode ser individual ou em lotes.

O *processo de serviço* na estação é descrito pelo tempo de processamento de *jobs*. As notações M e G indicam, respectivamente, um processo markoviano e genérico (ou genérico independente). Estações podem ter uma única máquina (servidor), ou várias máquinas. Cada máquina pode executar uma operação em um *job* individual, ou em lotes de *jobs*. Usaremos a notação GI/G/m para

denotar um sistema de fila única no qual o processo de chegada é de renovação (GI), os tempos de processamento são variáveis aleatórias *independentes e identicamente distribuídas* - *iid* (G), e o número de servidores no sistema é m.

Finalmente, *a fila de espera* na estação pode ter capacidade limitada ou ilimitada para o número de *jobs* na fila, geralmente determinada pelo espaço físico disponível. Uma vez alcançada, ela bloqueia a chegada de novos *jobs* na fila. A fila tem uma disciplina ou regra para ordenar os *jobs* esperando por processamento. Uma disciplina muito usual é a regra *primeiro-a-chegar primeiro-a-ser-servido* (*first-come first-serve* - FCFS).

O conjunto de nós, arcos e *jobs* compõe a rede de filas com as seguintes características: (i) número de estações (nós), (ii) sequência de operações (roteiros), e (iii) tipo de rede de filas: aberta, fechada e mista. Numa rede de filas *aberta* (*open queueing network* - *OQN*), os *jobs* entram na rede, recebem processamento em um ou mais nós, e eventualmente saem da rede. O número de *jobs* fluindo entre os nós é uma variável aleatória. Numa rede de filas *fechada* (*closed queueing network* - *CQN*), ao contrário, não há chegadas nem partidas externas de *jobs*. A taxa de partida de cada nó é uma variável aleatória mas o número de *jobs* fluindo entre os nós é fixo. Se a rede tiver múltiplas classes de *jobs*, podemos redefinir sub-redes abertas para algumas classes e sub-redes fechadas para outras. Neste caso, a rede de filas é chamada *mista*.

Os modelos de redes de filas analisados neste artigo assumem que o sistema atinja estado de equilíbrio ou *steady-state*. Os processos de chegada são probabilísticos com *iid* intervalos de tempos entre-chegadas nas estações. Os *jobs* podem pertencer a uma única classe ou a múltiplas classes, mas chegam individualmente nas estações. Não há limite no número de *jobs* em cada classe, mas os *jobs* não podem mudar de uma classe para outra. Os processos de serviço também são probabilísticos com *iid* tempos de processamento em cada estação. Cada estação pode ter uma ou mais máquinas idênticas, e cada máquina serve apenas um *job* por vez. *Jobs* não podem ser criados ou combinados nas estações. As filas de espera têm capacidade ilimitada, com disciplina FCFS. Todos os modelos discutidos neste artigo referem-se a OQN com estruturas acíclicas ou cíclicas e roteiros determinísticos ou probabilísticos. Outros modelos para os diversos casos não considerados neste exame podem ser encontrados na literatura discutida a seguir, e nas referências lá citadas.

Outros Exames na Literatura

Diversos exames dos modelos de avaliação de desempenho de redes de filas foram referidos no nosso primeiro artigo; entre eles, LEMOINE (1977), KOENIGSBERG (1982), DISNEY & KONIG (1985), BUZACOTT & YAO (1986), e SURI et al (1993).

BUZACOTT & SHANTHIKUMAR (1992, 1993), HSU et al (1993) e BITRAN & DASU (1992) examinaram ambos os modelos de avaliação de desempenho e os modelos de otimização de redes de filas. Buzacott e Shanthikumar realizaram uma extensa análise, orientada para o projeto de diferentes sistemas de manufatura, tais como linhas de fluxo, linhas de transferência automatizadas, *job shops*, FMS (*flexible manufacturing systems*) e sistemas multicelulares. Eles analisaram problemas de projeto ótimo e em particular, consideraram alguns modelos de otimização em *job shops* que não serão aqui tratados, tais como: alocação ótima de operadores nas estações, número ótimo de operadores no sistema, alocação ótima de *jobs* nas estações, e análise dos efeitos da diversidade de roteiros e tempos de processamento de *jobs*. HSU et al (1993) examinaram modelos de otimiza-

ção para FMS baseados em CQN; eles também sugeriram o uso de técnicas alternativas, como álgebra max-plus, conjuntos nebulosos e sistemas especialistas. Bitran e Dasu discutiram problemas estratégicos, táticos e operacionais de sistemas de manufatura, com uma atenção especial para os modelos de projeto e planejamento de *job-shops*. Como foi mencionado anteriormente, tal enfoque é estendido neste presente artigo.

Muitas abordagens para os modelos de otimização são baseadas nos *métodos de decomposição* (veja seção 3*) para avaliar medidas de desempenho de uma OQN. Mais recentemente, abordagens alternativas têm sido exploradas (modelos brownianos), baseadas nos teoremas do limite de tráfego-pesado, para avaliar medidas de desempenho e tratar modelos de otimização. Na seção 5 apresentamos um exemplo dessas abordagens (WEIN, 1990a), sem explorar desenvolvimentos adicionais neste tópico, uma vez que HARRISON & NGUYEN (1993) recentemente revisaram o estado-da-arte dos modelos brownianos para OQN com múltiplas classes.

No texto que segue, procuramos reservar os índices i e j para indicar cada estação, o índice k para indicar cada classe de produtos, e o índice *l* para indicar cada operação de uma classe numa estação. As notações E(x), V(x) e cx designam, respectivamente, a média, a variância, e o coeficiente quadrado de variação (*square coefficient of variation - scv*) (definido por $cx=V(x)/E(x)^2$) da variável aleatória x. Vetores e matrizes são indicados em negrito.

3. Modelos de Otimização de OQN

Na seção 3* examinamos modelos para avaliar o desempenho de uma OQN representando um sistema *job-shop*. Nesta seção examinamos modelos para projetar ou reprojetar uma OQN existente representando um *job-shop*. Naturalmente, se o projeto envolve selecionar uma configuração dentre um pequeno número de alternativas, então podemos utilizar os modelos anteriores para escolher a opção de melhor desempenho; caso contrário, precisamos de modelos baseados em técnicas de otimização para encontrá-la. BITRAN & DASU (1992) classificaram os modelos de otimização de OQN em:

(i) projeto ótimo
(ii) controle ótimo.

Modelos de *projeto ótimo* determinam o projeto ótimo do sistema para uma dada regra operacional (p.e., a disciplina FCFS). Modelos de *controle ótimo* determinam a regra operacional ótima para uma dada configuração do sistema. Neste trabalho revemos apenas os modelos de projeto ótimo. Para uma recente discussão dos modelos de controle ótimo baseados no movimento browniano, veja HARRISON & NGUYEN (1993).

Os três problemas SP1.1, SP2.1 e SP3.1 apresentados na seção 1 são exemplos de problemas de projeto ótimo. Conforme foi discutido na seção 1, diversas medidas de desempenho podem ser utilizadas, tais como o WIP, *leadtimes*, e taxa de produção. A seguir formulamos os problemas SP1.1 e SP2.1 escolhendo o WIP como medida de desempenho. Uma vez que o WIP e o *leadtime* são linearmente relacionados por meio da *lei de Little*, os algoritmos a serem descritos abaixo também se aplicam para o *leadtime*. Estudos similares utilizando a taxa de produção como medida de desempenho podem ser encontrados em BITRAN & SARKAR (1994a). Por simplicidade, adotamos a notação $L_j(.)$ e $W_j(.)$, ao invés de $E(L_j(.))$ e $E(W_j(.))$ adotada no primeiro artigo, para designar o número médio de *jobs* e o tempo médio de espera em fila e serviço na estação j. Sejam:

$\#\#\#_j$ taxa média de serviço de cada máquina

na estação j,

m_j número de máquinas idênticas na estação j,

$F_j(\#\#\#_j, m_j)$ custo de alocar a capacidade ($\#\#\#_j$, m_j) na estação j,

F_T orçamento disponível para a capacidade da rede,

$L_j(\#\#\#_1, m_1; \#\#\#_2, m_2; ...; \#\#\#_n, m_n)$ o número médio de *jobs* na estação j, como uma função da capacidade da rede,

v_j valor monetário médio de um *job* na estação j, independente de sua classe,

L_T limitante superior para o WIP da rede.

Lembramos que o WIP é um valor monetário médio do número médio de *jobs* na rede, definido como: $\#\#\#_{j=1,..,n}\ v_j\ L_j(\#\#\#_1, m_1;$ $\#\#\#_2, m_2; ...; \#\#\#_n, m_n)$. Cada valor monetário v_j, associado a um *job* na estação j, pode ser estimado usando-se experiência prática, ou como uma média ponderada proporcional à taxa média de chegada e ao tempo médio de espera de cada classe (o tempo médio de espera pode ser computado aproximadamente através de um procedimento proposto em ALBIN, 1986). Obviamente, se $v_j=1$ para todo j, então o WIP corresponde ao número médio de *jobs* na rede.

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

(SP1.1)

	minimizar	$\#\#\#_{j=1,..,n}\ F_j(\#\#\#_j, m_j)$
	sujeito a:	$\#\#\#_{j=1,..,n}\ v_j\ L_j(\#\#\#_1, m_1; \#\#\#_2, m_2; ...; \#\#\#_n, m_n)$ ### L_T
	com:	$(\#\#\#_j, m_j)\#\#\#P_j$, j=1,..,n.

onde P_j é um dado domínio das variáveis. Similarmente, o problema de WIP ótimo SP2.1 é o problema de determinar a capacidade $(\#\#\#_j, m_j)$ 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:

(SP2.1)

	minimizar	$\#\#\#_{j=1,..,n}\ v_j\ L_j(\#\#\#_1, m_1; \#\#\#_2, m_2; ...; \#\#\#_n, m_n)$
	sujeito a:	$\#\#\#_{j=1,..,n}\ F_j(\#\#\#_j, m_j) = F_T$
	com:	$(\#\#\#_j, m_j)\#\#\#P_j$, j=1,..,n.

Diversos autores apresentaram métodos de solução para os dois problemas acima. A seguir revemos alguns desses procedimentos. Com a finalidade de apresentá-los de uma forma mais estruturada, adotaremos a notação sugerida em BITRAN & DASU (1992) que designa cada instância do problema por ###/###/###/###, onde ### ###{SP1.1, SP2.1, SP3.1}, ######{J, G}, ######{S, M} e ######{R, N}. O símbolo ### indica o tipo de problema, ### indica se o problema é aplicado a uma rede de Jackson (J) ou a uma rede genérica (G), ### indica se as estações têm apenas uma única máquina (S) ou múltiplas máquinas (M), e ### indica a variável de decisão do modelo: taxa média de serviço (R) ou número de máquinas (N) em cada estação.

Os problemas SP1.1 e SP2.1 são considerados nas seções 4 e 5. Na seção 4 revemos modelos e métodos de solução para as chamadas *redes de Jackson* (OQN com processos de chegada e serviço markovianos) e na seção 5, para as *redes de Jackson generalizadas* (OQN com processos de chegada e

serviço genéricos). Para maiores detalhes das redes de Jackson e redes de Jackson generalizadas, veja as seções 3* e 4*. Conforme salientamos anteriormente, este presente exame analisa apenas modelos de otimização em OQN. Exemplos de modelos de otimização em CQN podem ser encontrados, por exemplo, em SHANTHIKUMAR & YAO (1987, 1988), DALLERY & STECKE (1990) e CALABRESE (1992).

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

Nas redes de Jackson podemos decompor cada estação j como um sistema estocasticamente independente. Desta maneira, L_j em SP1.1 e SP2.1 torna-se uma função apenas de $###_j$ e m_j, ao invés de uma função de $###_1$, m_1; $###_2$, m_2; ...; $###_n$, m_n.

4.1 Modelos ./J/./R

KLEINROCK (1964, 1976) inicialmente estudou o problema de minimizar o número médio de *jobs* em redes de Jackson com um servidor e classe única. Consideremos novamente os dados de entrada da seção 4.1* com $m_j=1$, j=1,..,n. Kleinrock escolheu as taxas de serviço $###_j$ como variáveis de decisão, e assumiu que o custo F_j fosse proporcional a $###_j$ em cada estação j: $F_j(###_i)=f_j###_j$, onde f_j é o custo unitário da capacidade na estação j. Aplicando a lei de Little ($Lq_j=###_jWq_j$) em (7*) e adicionando a carga ofertada ($###_j=###_j/###_j$), obtemos o número médio de *jobs* em um sistema M/M/1 definido como: $L_j(###_i)=###_j/(###_j-###_j)$, onde $###_j$ é computado conforme (1*). O modelo SP2.1/J/S/R é então formulado por:

(SP2.1/J/S/R) minimizar $###_{j=1,..,n} L_j(###_i)$
sujeito a: $###_{j=1,..,n} f_j ###_j = F_T$
com: $###_j ### 0$, j=1,..,n.

Introduzindo o multiplicador de Lagrange y associado à restrição de igualdade em SP2.1/J/S/R, obtemos a função lagrangeana: $###_{j=1,..,n}L_j(###_i) + y (###_{j=1,..,n}f_j###_j - F_T)$. Ao derivarmos essa função com relação à cada $###_j$, obtemos a solução ótima $###_j$*, j=1,..,n, de SP2.1/J/S/R, dada em forma fechada por:

$$###_j^* = ###_j + [###(f_j###_j) / ###_{i=1,..,n}###(f_i###_i)] [(F_T - ###_{i=1,..,n}f_i###_i) / f_j] \quad (1)$$

Note que se o custo unitário da capacidade for igual em todas as estações, (1) primeiro aloca capacidade na estação j suficiente para satisfazer a taxa média de chegada, e depois aloca capacidade na estação j na proporção da raiz quadrada da sua taxa média de chegada. Conforme BITRAN & DASU (1992) observaram, cinco condições são satisfeitas no modelo acima:

(i) $L_j(###_j)$ é uma função convexa em $###_j$ (o número médio de *jobs* na estação j é uma função convexa da capacidade da estação j),
(ii) $L_j(###_j)$ não depende de $###_i$, i###j, i=1,..,n (adições de capacidade em outras estações têm nenhum efeito no número médio de *jobs* da estação j),
(iii) $###_j$ é contínuo (as variáveis de

decisão são variáveis contínuas),

(iv) $F_j(\#\#\#_j)$ é uma função convexa em $\#\#\#_j$ (o custo de capacidade da estação j é uma função convexa da capacidade da estação j),

(v) $L_j(\#\#\#_j)$ (e $W_j(\#\#\#_j)$) pode ser expresso em forma fechada.

As condições (i)-(iv) reduzem o modelo SP2.1/J/S/R a um programa convexo que pode ser resolvido otimamente via métodos de ótimo-local (BAZARAA et al., 1993). A condição (v) permite que a solução do problema seja em forma fechada. Estas condições serão extensivamente utilizadas ao longo da seções 4 e 5.

Podemos formular SP2.1/J/M/R exatamente como SP.2.1/J/S/R, onde $L_j(\#\#\#_j)$ é redefinido para filas $M/M/m_j$ (compare com (7*)). HAREL & ZIPKIN (1987) mostraram que o tempo médio de espera $W_j(\#\#\#_j)$ (e o número médio de *jobs* $L_j(\#\#\#_j)$) em uma fila $M/M/m_j$ também é uma função convexa em $\#\#\#_j$. Assim, as condições (i)-(iv) também são satisfeitas e SP2.1/J/M/R também pode ser reduzido a um programa convexo.

Os modelos SP1.1/J/S/R e SP1.1/J/M/R podem ser definidos e analisados de maneira similar.

4.2 Modelos ./J/M/N

A seguir discutimos os modelos SP1.1/J/M/N e SP2.1/J/M/N. Note agora que as variáveis de decisão são inteiras, correspondendo ao número de máquinas em cada estação. BOXMA et al. (1990) apresentaram um algoritmo heurístico e outro exato para resolvê-los, respectivamente. A rede de manufatura é representada por uma OQN de Jackson com múltiplos servidores, múltiplas classes, e diferentes roteiros determinísticos para cada classe. (veja seções 4.2* e 5.3*). Considere novamente os dados de entrada descritos na seção 5.3* com $ca_k=1$ e $cs_k=1$ para cada classe k, e $m_j\#\#\#1$ para cada estação j.

Ao agregarmos todas as classes em uma única classe, obtemos cada estação j descrita pelos três parâmetros $\{m_j, \#\#\#_j, \#\#\#_j\}$ (veja seção 4.2*). KELLY (1979) mostrou que a distribuição de equilíbrio do número de *jobs* na rede pode ser expressa na forma de produto, e que cada estação j em estado de equilíbrio se comporta como um sistema $M/M/m_j$. Aplicando a lei de Little em (7*) e somando a carga ofertada, obtemos o número médio de *jobs* L_j em função de m_j, $\#\#\#_j$ e $\#\#\#_j$, dado por:

$$L_j(m_j, \#\#\#_j, \#\#\#_j) = \{\#\#\#_j/\#\#\#_j m_j \ (\#\#\#_j/\#\#\#_j)^{mj} \ \#\#\#(0) / [m_j! \ (1 - \#\#\#_j/\#\#\#_j m_j)^2]\} + \#\#\#_j/\#\#\#_j \qquad (2)$$

onde,

$$###(0) = \{###_{t=0,..,mj-1} (###_j/###_j)^t / t! + (###_j/###_j)^{mj} / [m_j! (1- ###_j/###_j m_j)]\}^{-1}$$

BOXMA et al (1990) consideraram m_j, j=1,...,n, como variável inteira nos modelos SP1.1/J/M/N e SP2.1/J/M/N, e observaram que $L_j(m_j, ###_j, ###_j)$ em (2), denotado simplesmente por $L_j(m_j)$, é uma função convexa e decrescente em m_j (as condições (i) e (ii) são satisfeitas). Eles escolheram o WIP como medida de desempenho da rede. Note que esta análise pode ser facilmente estendida para o *leadtime*, dado que o WIP e o *leadtime* são linearmente relacionados. Seja **m** o vetor de variáveis (m_1, m_2,..., m_n). O WIP da rede, L(**m**), é dado por:

$$L(\mathbf{m}) = ###_{j=1,..,n} v_j L_j(m_j) \quad (3)$$

A escolha de m_j em cada estação deve satisfazer a condição $###_j###1$ para evitar a instabilidade do sistema. Seja ###z### denotando o maior inteiro menor ou igual a z. Usando (2*), segue que m_j deve ser um número inteiro maior ou igual ao limitante inferior m_j^0, definido por:

$$m_j^0 = ######_j###_j### + 1 \quad (4)$$

Modelo SP1.1/J/M/N

No modelo SP1.1/J/M/N queremos encontrar a solução de custo mínimo satisfazendo um nível de WIP menor ou igual a um limite especificado L_T, com $L_T ### L(\mathbf{m}^0)$. Seja $F_j(m_j)$ o custo de alocar m_j máquinas na es-tação j, definido como uma função convexa não-decrescente de m_j (a condição (iv) é satisfeita). Utilizando (3) e (4), obtemos o problema de WIP desejado (também cha-mado *problema de alocação de servidores*):

(SP1.1/J/M/N)	minimizar	$F(\mathbf{m}) = ###_{j=1,..,n} F_j(m_j)$
	sujeito a:	$L(\mathbf{m}) ### L_T$
	com:	$m_j ### m_j^0$, m_j inteiro, j=1,..,n.

Note que, dado que SP1.1/J/M/N é um programa convexo com variáveis inteiras, o uso de análise marginal não leva necessariamente à otimalidade (a condição (iii) não é satisfeita). Seja $PI_j(m_j)$ um *índice de prioridade* definido como o quociente entre o aumento de custo e a redução de WIP na estação j, dado por:

$$PI_j(m_j) = ###F_j(m_j+1) / -v_j ###L_j(m_j+1) \quad (5)$$

onde

$###F_j(m_j+1)=F_j(m_j+1)-F_j(m_j)###0$ e

$###L_j(m_j+1)=L_j(m_j+1)-L_j(m_j)###0$.

PI_j é resultado da análise marginal de F_j e L_j. BOXMA et al (1990) apresentaram um simples algoritmo heurístico (algoritmo 1) baseado no *método guloso* para resolver o problema SP1.1/J/M/N (veja também a abordagem em SUNDARRAJ et al (1994) para um problema similar). O algoritmo começa com a menor alocação de máquinas possível (4) para cada estação. Em cada iteração, ele adiciona uma máquina na estação com o mínimo índice de prioridade (5). O algoritmo termina assim que a adição de uma máquina resultar numa alocação factível.

Algoritmo 1

1. Comece com a alocação $m_j=m_j^0$, j=1,..,n. Esta solução é infactível ($L(\mathbf{m}^0)$###L_T) e seu custo $F(\mathbf{m}^0)$ é menor do que o custo mínimo de SP1.1/J/M/N.
2. Em cada iteração, atualize o custo F(**m**), o WIP L(**m**) (usando (2) e (3)) e $PI_j(m_j)$ (usando (5)). Adicione uma máquina na estação j* que resultar no menor quociente PI_{j*} (estratégia gulosa), dado por:

$$PI_{j*} = \text{mínimo } \{PI_j(m_j), j=1,..,n\} \qquad (6)$$

3. Pare assim que L(**m**) atingir o limite L_T (solução factível).

Note que a estação j* escolhida em (6) produz o menor aumento em F(**m**) por unidade de redução em L(**m**), indicado por PI_{j*}. Devido à convexidade de F_j e L_j, temos que:

$$\#\#\#F_j(m_j+1) \,/\, -v_j \;\#\#\#L_j(m_j+1) \;\#\#\# \;\#\#\# F_j(m_j) \,/\, -v_j \;\#\#\#L_j(m_j) \qquad (7)$$

Um resultado interessante que decorre de (7) é que podemos verificar a qualidade da solução heurística gerada pelo algoritmo 1, apenas comparando as soluções geradas nas duas últimas iterações. Seja p a última iteração, e $\mathbf{m}^1$, $\mathbf{m}^2$,..., $\mathbf{m}^{p-1}$, $\mathbf{m}^p$ as soluções geradas em cada iteração. Obviamente, $\mathbf{m}^{p-1}$ é infactível e $\mathbf{m}^p$ é factível. Denotemos por $\mathbf{m}^*$ a solução ótima de SP1.1/J/M/N. BOXMA et al. (1990, teoremas 1 e 2) mostraram que:

$$F(\mathbf{m}^{p-1}) \;\#\#\# F(\mathbf{m}^*) \;\#\#\# F(\mathbf{m}^p)$$

e portanto, $F(\mathbf{m}^{p-1})$ e $F(\mathbf{m}^p)$ são limitantes para o valor da solução ótima. Experimentos computacionais utilizando duas redes de manufatura da vida real resultaram num erro relativo de 5% entre $F(\mathbf{m}^p)$ e $F(\mathbf{m}^{p-1})$. Estas experiências sugerem que o algoritmo 1 gera uma alocação suficientemente próxima da alocação ótima de SP1.1/J/M/N.

Modelo SP2.1/J/M/N

No modelo SP2.1/J/M/N queremos alocar (ou realocar) máquinas de maneira a otimizar uma medida de desempenho, por exemplo, minimizar o WIP na rede. Assumimos que um total de M máquinas homogêneas deve ser alocado às estações, onde M$\#\#\#\#\#\#_{j=1,..,n} m_j^0$. Esta situação ocorre por exemplo no projeto de FMS, no qual podemos ter máquinas iguais desempenhando operações diferentes ao instalarmos ferramentas diferentes. Utilizando (3) e (4) obtemos o problema de WIP ótimo (também chamado de *problema de realocação de servidores*):

$$\begin{array}{lll} \text{(SP2.1/J/M/N)} & \text{minimizar} & L(\mathbf{m}) \\ & \text{sujeito a:} & \#\#\#_{j=1,..,n}\, m_j = M \\ & \text{com:} & m_j \;\#\#\#\; m_j^0, \;\; m_j \text{ inteiro, } j=1,..,n. \end{array}$$

Novamente, temos um programa convexo com variáveis inteiras (as condições (i), (ii) e (iv) são satisfeitas mas a condição (iii) é violada), e o uso de análise marginal pode não produzir a solução ótima de SP2.1/J/M/N. Seja agora $PI_j(m_j)$ um índice de prioridade definido como a redução de WIP por unidade de máquina na estação j, dado por:

$$PI_j(m_j) = -v_j \, \#\#\#L_j(m_j+1) \quad (8)$$

onde, $\#\#\#L_j(m_j + 1) = L_j(m_j + 1) - L_j(m_j) \#\#\# 0$, conforme seção anterior.

BOXMA et al (1990) apresentaram um simples algoritmo (algoritmo 2), similar ao algoritmo 1, também baseado no método guloso, para resolver SP2.1/J/M/N. O algoritmo começa com a menor alocação de máquinas possível (4) para cada estação. Em cada iteração ele adiciona uma máquina na estação com o máximo índice de prioridade (8). O algoritmo termina quando todas as M máquinas tiverem sido alocadas.

Algoritmo 2

1. Comece com a alocação $m_j=m_j^0$, j=1,..,n. Esta solução é infactível ($\#\#\#_{j=1,..,n}m_j^0$ ###M) e seu WIP L($\mathbf{m}^0$) é maior do que o WIP mínimo de SP2.1/J/M/N.
2. Em cada iteração, atualize o WIP L(**m**) (usando (2) e (3)) e $PI_j(m_j)$ (usando (8)). Adicione uma máquina na estação j* que resultar no maior PI_{j*} (estratégia gulosa), dado por:

$$PI_{j*} = \text{máximo } \{PI_j(m_j), j=1,..,n\} \quad (9)$$

3. Pare assim que o número total de máquinas alocadas atingir o limite M (solução factível).

Note que a estação j* escolhida em (9) produz a maior redução em L(**m**) por unidade de máquina, indicada por PI_{j*}. Devido à convexidade de L_j, temos que:

$$v_j \, \#\#\#L_j(m_j+1) \#\#\# v_j \, \#\#\#L_j(m_j) \quad (10)$$

Usando (10), BOXMA et al. (1990, teorema 3) provaram que o algoritmo 2 é exato e termina com a solução ótima de SP2.1/J/M/N (apesar da condição (iii) não ser satisfeita). Além disso, esta solução é gerada num intervalo de tempo limitado por uma função polinomial no número de estações da rede, ou seja, em O(Mn).

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

Nesta seção estudamos os modelos SP1.1/G/S/R, SP2.1/G/S/R (BITRAN & TIRUPATI, 1989a, BITRAN & SARKAR, 1994a, e WEIN, 1990a), SP1.1/G/M/R com variáveis discretas (BITRAN & TIRUPATI, 1989b), e SP1.1/G/M/N e SP2.1/G/M/N (VAN VLIET & RINNOOY KAN, 1991).

5.1 Modelos ./G/./R

Inicialmente, apresentamos dois algoritmos introduzidos por BITRAN & TIRUPATI (1989a) para os modelos SP1.1/G/S/R e SP2.1/G/S/R. Estes algoritmos podem ser facilmente estendidos para tratar os modelos SP1.1/G/M/R e SP2.1/G/M/R. A rede de manufatura é representada por uma OQN com múltiplas classes e roteiros determinísticos para cada classe (veja seção 5.3*). Na seção anterior, medidas de desempenho como o WIP em (3) foram facilmente avaliadas devido aos resultados exatos das redes de Jackson. Na falta de métodos exatos para as redes de Jackson generalizadas, o método aproximado de decomposição discutido na seção 5.3* é utilizado para estimar (aproximadamente) os parâmetros de variabilidade em cada estação. Considere novamente os dados de entrada da seção 5.3* com $m_j=1$ para toda estação j. O passo 1 do método de decomposição resulta no sistema de equações (13*) mais (29*), (31*) e (32*), representados a seguir simplesmente por:

###(###, **ca**, ###, **cs**) = 0 (11)

onde os vetores ###, **ca**, ### e **cs** denotam { $###_j$, ca_j, $###_j$, cs_j} para todo j. Aplicando a lei de Little em (16*) e adicionando a carga oferta-da, obtemos o número médio de *jobs* L_j co-mo uma função de $###_j$, ca_j, $###_j$ e cs_j, dada por:

$$L_j(###_j, ca_j, ###_j, cs_j) = (###_j/###_j)^2 (ca_j + cs_j)\, g(###_j, ca_j, ###_j, cs_j) / 2 (1 - ###_j/###_j) + ###_j/###_j \quad (12)$$

onde $g(###_j, ca_j, ###_j, cs_j)$ é definido conforme (16*). Uma vez que L_j é uma função de $###_j$, ca_j, $###_j$ e cs_j em (12), obtemos L_j como uma função de ###, **ca**, ### e **cs**. Bitran e Tirupati consideraram cada capacidade $###_j$, j=1,...,n, como variável de decisão contínua (portanto, a condição (iii) é satisfeita), admitindo que capacidade adicional possa ser incorporada na estação em pequenos incrementos, quando comparados com a capacidade total (lembre-se que estamos supondo uma única máquina em cada estação). Para um dado ###, (11) e (12) sugerem que mudanças na capacidade ### resultem em mudanças em **ca** e **cs**. Assim, L_j é uma função de $###_1$, $###_2$, ..., $###_n$. Entretanto, esta relação funcional é complexa, dado que o sistema de equações em (11) é não-linear e não é fácil de ser analisado.

Bitran e Tirupati assumiram que **ca** e **cs** sejam independentes de mudanças da capacidade ###. Dessa maneira, L_j não depende de $###_i$, i###j (condição (ii) satisfeita). Eles argumentaram que ao variarmos ###, a média e a variância do tempo de serviço variam na mesma proporção e portanto, **cs** mantém-se aproximadamente constante. Além disso, a sensibilidade de **ca** a mudanças de ### parece ser pequena, à medida que aumentamos o número de classes e a proporção da demanda devida por cada classe diminui (veja os resultados numéricos em BITRAN & TIRUPATI (1988) e a discussão em WHITT (1988)). A consequência dessas hipóteses é que podemos inicialmente resolver o sistema (11) para uma dada capacidade, e em seguida tratar os **ca** obtidos como parâmetros conhecidos em (12). Sob estas suposições, Bitran e Tirupati também mostraram que $L_j(###_j, ca_j, ###_j, cs_j)$ em (12) é uma função convexa em $###_j$, denotada agora simplesmente por $L_j(###_j)$ (condição (i) é satisfeita). Seja ### o vetor de variáveis. O WIP em (3) pode ser reescrito como (compare com (3)):

$$L(###) = ###_{j=1,..,n}\, v_j\, L_j(###_j) \quad (13)$$

Finalmente, designamos por $###_j^0$ um limitante inferior da capacidade na estação j. Note que este limitante deve satisfazer a condição $###_j$###1 para evitar a instabilidade do sistema:

$$###_j^0\ ###\ ###_j \quad (14)$$

Modelo SP1.1/G/S/R

Similarmente à seção 4.2, seja L_T uma meta para o nível de WIP da rede, tal que L_T ###L($###^0$). Seja também $F_j(###_j)$ o custo de alocar capacidade $###_j$ na estação j, definido como uma função convexa não-decrescente e diferenciável de $###_j$ (a condição (iv) é satisfeita). Utilizando (13) e (14), obtemos o seguinte problema de programação convexa:

(SP1.1/G/S/R) minimizar $F(###) = ###_{j=1,..,n}\, F_j(###_j)$
sujeito a: $L(###)\ ###\ L_T$

com: $###_j$ ### $###_j^0$, j=1,..,n.

BITRAN & TIRUPATI (1989a) apresentaram um algoritmo heurístico (algoritmo 3) para resolver SP1.1/G/S/R e gerar curvas de *tradeoff* entre F(###) e L(###). Seja $PI_j(###_j)$ um índice de prioridade, agora definido como o quociente do aumento marginal de custo pela redução marginal de WIP na estação j, dado por:

$PI_j(###_j) = ###F_j(###_j)/######_j$ / $-v_j$ ### $L_j(###_j)/######_j$ (15)

O algoritmo 3 é similar ao algoritmo 1 da seção 4.2. Seja ### um incremento de capacidade em cada iteração, previamente especificado. Começamos com a capacidade inicial satisfazendo (14) para todas as estações. Em cada iteração, aumentamos de ### a capacidade da estação com o mínimo índice de prioridade em (15). O procedimento é repetido até que a meta L_T seja alcançada.

Algoritmo 3

1. Comece com a alocação $###_j=###_j^0$ (suficientemente pequena) e compute os valores ca_j e cs_j (usando (11)) para cada estação j, j=1,..,n. Esta solução é infactível ($L(###^0)###L_T$) e seu custo $F(###^0)$ é menor do que o custo mínimo de SP1.1/G/S/R.
2. Em cada iteração, atualize o custo F(###), o WIP L(###) (usando (12) e (13)) e o quociente $PI_j(###_j)$ (usando (15)). Adicione a ca-pacidade ### na estação j* que resultar no menor PI_{j*} (estratégia gulosa), dado por:
   PI_{j*} = mínimo $\{PI_j(###_j), j=1,..,n\}$ (16)
3. Pare assim que L(###) atingir o limite L_T (solução factível).

À medida que escolhemos valores menores para ###, o algoritmo 3 gera curvas de *tradeoff* mais precisas. BITRAN & TIRUPATI (1989a, proposição 2) mostraram que no limite ######0, o algoritmo 3 resolve otimamente SP1.1/G/S/R (lembre-se que assumimos que todas as condições (i)-(iv) são satisfeitas), e o $PI_j(###_j)$ obtido na última iteração corresponde ao multiplicador dual associado à restrição de WIP da estação j.

BITRAN & TIRUPATI (1989a, proposição 3) também apresentaram um limitante para o erro do valor da solução aproximada obtida pelo algoritmo 3. Suponha que o algoritmo 3 encontre uma solução factível após p iterações, denotada por $###^p$, e seja ###* a solução ótima de SP1.1/G/S/R. Temos que:

0 ### F($###^p$) - F(###*) ### ($\mathbf{L}_T$ - L($###^p$))/PI_{j*}^p + ### (17)

onde ### = ### $###_{i=1,p}(1 - PI_{j*}^i/PI_{j*}^p)$, e PI_{j*}^i é o quociente obtido em (16) na i-ésima iteração, i=1,..,p. Experiências computacionais com ###=0.1 aplicadas em um exemplo real com 13 estações e 10 classes resultaram num erro relativo de 0.6% entre F($###^p$) e F(###*). Esse erro é aceitável em muitas situações práticas. Essas experiências também indicaram que a suposição anterior de que **ca** e **cs** sejam independentes de mudanças de ### é razoável (observe no algoritmo 3 que **ca** e **cs** permanecem constantes ao longo das iterações). Como ilustração, ao reduzir o WIP de um valor inicial de 70000 para 30000, a maior variação encontrada no valor de **ca** foi de 3%. Esta variação foi obtida atualizando-se os valores de **ca** e **cs** conforme (11) para a configuração final da rede.

Um refinamento no algoritmo 3 é atualizar **ca** em cada iteração conforme (11). De

fato, BITRAN & SARKAR (1994b) exploraram esta alternativa. Quando **ca** não é mais considerado independente da capacidade, não há garantia de que SP1.1/G/S/R seja um programa convexo. Não sabemos se L em (13) permanece convexo em ### porque agora **ca** varia com a variação de ###, conforme (11). Assim, este procedimento alternativo pode não convergir para a solução ótima, ou mesmo não convergir para uma solução factível. Apesar disso, BITRAN & SARKAR (op.cit.) mostraram que o procedimento converge, sob certas condições para os dados iniciais.

Modelo SP2.1/G/S/R

A seguir, analisamos o problema de redistribuir capacidade existente nas estações de maneira a minimizar o WIP na rede. Esta redistribuição faz sentido em redes com capacidade homogênea, isto é, recursos que possam ser compartilhados por diferentes estações (e.g., mão-de-obra). Seja $###_j^1$ a capacidade inicial existente na estação j, tal que $###_j^1 ######_j^0$. Utilizando (13) e (14), temos:

(SP2.1/G/S/R) minimizar L(###)
sujeito a: $###_{j=1,..,n}\ ###_j = ###_{j=1,..,n} ###_j^1$
com: $###_j\ ###\ ###_j^0$, j=1,..,n.

BITRAN & TIRUPATI (1989a) apresentaram um algoritmo heurístico (algoritmo 4), também baseado no método guloso, para resolver SP2.1/G/S/R (novamente, as condições (i)-(iv) são satisfeitas). Sejam ### definido conforme anteriormente e $PI_j(###_j)$ definido agora como a redução marginal de WIP na estação j, dado por:

$$PI_j(###_j) = -v_j\ ###L_j(###_j)/######_j \quad (18)$$

Algoritmo 4

1. Comece com a alocação $###_j=###_j^1$ (solução factível) e compute os valores ca_j e cs_j (usando (11)) para cada estação j, j=1,..,n. Defina J_0 como o conjunto das estações disponíveis, J_1 como o conjunto das estações cuja capacidade for aumentada e J_2 como o conjunto das estações cuja capacidade for reduzida. Inicialmente $J_0=\{1,2,..,n\}$, e J_1 e J_2 são vazios. Compute ### tal que:

$$PI_j(###_j)\ (###_j + ###_j) = \text{máximo}\ \{PI_j(###_j),\ j###J_0\} \quad (19)$$

2. Em cada iteração, atualize o WIP L(###) (usando (12) e (13)) e $PI_j(###_j)$ (usando (18)). Encontre a estação j_1 que resultar no menor PI_{j1} dado por:

$$PI_{j1} = \text{mínimo}\ \{PI_j(###_j),\ j###J_0\} \quad (20)$$

e a estação j_2 que resultar no maior PI_{j2} dado por:

$$PI_{j2} = \text{máximo}\ \{PI_j(###_j),\ j###J_0\} \quad (21)$$

2a. Se $j_1###J_1$, então faça $J_0###J_0-\{j_1\}$.
2b. Se $j_2###J_2$, então faça $J_0###J_0-\{j_2\}$.
2c. Se $j_1###J_1$ e $j_2###J_2$, então defina $###_1=$ $\min\{###,\ ###_{j1}-###_{j1}-###_{j1}\}$ e faça $###_{j1}######_{j1}-$ $###_1$, $###_{j2}######_{j2}+###_1$, $J_1###J_1 ###\{j_2\}$ e $J_2###J_2###\{j_1\}$.

3. Pare se J_0 for nulo ou unitário, ou $PI_{j1}=PI_{j2}$.

Note que, em cada iteração, (20) e (21) correspondem às estações que produzem a maior e a menor redução marginal em L(###), respectivamente. A expressão (19) junto com $###_1$ garante que a solução gerada pelo algoritmo 4 satisfaz $###_j######_j+###_j$, j=1,..,n. Portanto, ela satisfaz (14) e é factível. BITRAN & TIRUPATI (1989a, remark) mostraram que no limite ######0, esta solução é ótima para SP2.1/G/S/R (lembre-se de que assumimos todas as condições (i)-(iv) satisfeitas), e todos os $PI_j($ $###_j$) produzidos na última iteração têm o mesmo valor. Similarmente à seção anterior, cada $PI_j(###_j)$ pode ser interpretado como o multiplicador dual associado às restrições de capacidade na estação j. Ele representa a taxa de redução do WIP em relação a adições marginais de capacidade nesta estação.

BITRAN & TIRUPATI (1989a, proposição 4) também apresentaram um limitante para o erro do valor da solução aproximada gerada pelo algoritmo 4. Seja $###^p$ a solução heurística gerada na última iteração p, e ### * a solução ótima de SP2.1/G/S/R. Então:

$$0 \text{ ### } F(###^p) - F(###^*) \text{ ### } n \text{ ### } PI_{j2}^p \quad (22)$$

onde PI_{j2}^p é o índice de prioridade obtido em (21) na última iteração p. Observe em (22) que a solução $###^p$ é ótima no limite ######0. Experiências computacionais com ###=0.02 aplicadas no mesmo exemplo da seção anterior resultaram num erro relativo menor que 2% entre $F(###^p)$ e F(###*), indicando que o algoritmo 4 é uma boa aproximação para SP2.1/G/S/R. Bitran e Tirupati reportaram um resultado interessante deste exemplo: o WIP foi reduzido de um valor inicial de 70000 para um valor final perto de 40000 apenas redistribuindo a capacidade inicial da rede (note no entanto que eles supuseram que a capacidade de cada estação fosse com-pletamente transferível para as outras esta-ções). Com a finalidade de testar a hipótese da independência de **ca** em relação a alte-rações da capacidade, eles recomputaram **ca** conforme (11) para a configuração final da rede (lembre-se que o algoritmo 4 mantém **ca** e **cs** fixados durante as iterações). A maior variação de **ca** foi em torno de 3%.

Note que os algoritmos 2 e 4 (problemas de WIP ótimo) ajudam a balancear o sistema de manufatura, enquanto os algoritmos 1 e 3 (problemas de WIP desejado) eficientemente adicionam recursos ao sistema. Podemos gerar curvas de *tradeoff* entre o capital de trabalho (WIP) e o capital de investimento, primeiramente, aplicando o algoritmo 4 para a configuração original do sistema e então, utilizando a solução obtida $###^p$ como capaci-dade inicial no algoritmo 3 (i.e., $###^0$ $######^p$, onde $###^0$ é a capacidade inicial no passo 1 do algoritmo 3). Para uma experiência computacional e análise de curvas de *tradeoff*, veja por exemplo BITRAN & TIRUPATI (1989a) e BITRAN & MORABITO (1995a).

WEIN (1990a) analisou o problema SP2.1/G/S/R numa OQN GI/G/1 de classe única com todos os *jobs* percorrendo um roteiro probabilístico. Partindo do modelo browniano proposto por HARRISON & WILLIAMS (1987), que é baseado na aproximação de tráfego-pesado de REIMAN (1984), Wein obteve o número médio de *jobs* na estação j (em equilíbrio), dado por:

$$L_j(###_j) = ###_j / 2(###_j - ###_{0j}) \quad (23)$$

onde

$$###_j=###_{0j}ca_j+###_jcs_j+###_{i=1,..,n}###_i-q_{ij}(cs_iq_{ij}+1-q_{ij}).$$

Note que (23) não deriva dos métodos aproximados de decomposição discutidos na seção 5*, ao contrário da expressão (12). Além disso, (23) é válida somente se uma certa condição, denominada *skew-symmetry*, for satisfeita (veja a expressão (4) em WEIN, 1990a). Considere novamente a restrição de orçamento utilizada em

KLEINROCK (1964) e discutida na seção 4.1. Supondo-se que a condição de *skew-symmetry* seja satisfeita e utilizando-se (23), o modelo SP2.1/G/S/R pode ser formulado por:

(SP2.1/G/S/R') minimizar $\#\#\#_{j=1,..,n} L_j(\#\#\#_i)$
sujeito a: $\#\#\#_{j=1,..,n} f_j \#\#\#_j = F_T$
com: $\#\#\#_j \# \# \# 0$, j=1,..,n.

Após derivar a função lagrangeana deste problema, Wein obteve a solução em forma fechada $\#\#\#_j^*$, j=1,..,n, dada por (compare com a expressão (1)):

$$\#\#\#_j^* = \#\#\#_j + [\#\#\#(f_j\#\#\#_j) / \#\#\#_{i=1,..,n}\#\#\#(f_i\#\#\#_i)] [(F_T - \#\#\#_{i=1,..,n} f_i\#\#\#_i) / f_j] \quad (24)$$

Wein observou que a condição de *skew-symmetry* é satisfeita para as redes de Jackson (i.e., sistemas M/M/1 com ca_j=1 e cs_j=1, j=1,..,n), e que (24) se reduz a (1), que é a solução ótima de SP2.1/J/S/R'. Note que se f_j for igual em todas as estações, então (24) primeiro aloca capacidade na estação j apenas para compensar $\#\#\#_j$, e depois aloca capacidade na estação j na proporção da raiz quadrada do parâmetro $\#\#\#_j$.

Wein apresentou experiências computacionais com um exemplo simples de uma OQN satisfazendo a condição de *skew-symmetry*. Esses resultados mostraram que (24) produz uma solução muito próxima da solução ótima gerada mediante simulação. Embora (24) seja derivada sob condições de tráfego-pesado ($\#\#\#_j\#\#\#0.9$), ela também pode produzir boas aproximações para baixas intensidades de tráfego. Uma questão importante é investigar a qualidade da solução gerada por (24) em situações em que a condição de *skew-symmetry* não é satisfeita.

5.2 Modelo SP1.1/G/M/R com Variáveis Discretas

BITRAN & TIRUPATI (1989b) apresentaram um algoritmo para o modelo SP1.1/G/M/R com *alternativas discretas* para mudança de capacidade em cada estação. Os *jobs* pertencem a múltiplas classes e cada classe segue um roteiro determinístico, conforme a seção 5.3*. Similarmente ao que foi feito na seção 5.1, o sistema de equações (19*) mais (29*), (31*) e (32*) do passo 1 do método de decomposição é descrito abaixo simplesmente por:

$$\#\#\#(\mathbf{m}, \#\#\#, \mathbf{ca}, \#\#\#, \mathbf{cs}) = 0 \quad (25)$$

onde os vetores **m**, ###, **ca**, ### e **cs** denotam respectivamente os parâmetros $\{m_j, \#\#\#_j, ca_j, \#\#\#_j, cs_j\}$ para todas as estações j, j=1,..,n. Aplicando a lei de Little em (21*) e adicionando a carga ofertada, obtemos o número médio de *jobs* na estação j, dado por:

$$L_j(m_j, \#\#\#_j, ca_j, \#\#\#_j, cs_j) = \#\#\#_j (ca_j + cs_j) Lq_j(M/M/m_j) / 2 + \#\#\#_j/\#\#\#_j \quad (26)$$

onde $Lq_j(M/M/m_j)$ denota o número médio de *jobs* em fila num sistema $M/M/m_j$. Note que $Lq_j(M/M/m_j)$ corresponde ao primeiro termo do lado direito de (2). Similarmente a (12), $L_j(m_j, \#\#\#_j, ca_j, \#\#\#_j, cs_j)$ em (26) é função de m_j, $\#\#\#_j$, ca_j, $\#\#\#_j$ e cs_j satisfazendo (25) (para outras aproximações de L_j, veja p.e. WHITT, 1993). Ao invés de escolher **m** ou ### como as variáveis de decisão de modelo, Bitran e Tirupati

consideraram um número finito de alternativas para mudança de capacidade em cada estação. Seja n_j o número total de alternativas para a estação j. Para cada alternativa k, k=1,..,n_j, são dados:

m_{jk} número de máquinas idênticas da estação j na alternativa k,

$\#\#\#_{jk}$ taxa média de serviço de cada máquina da estação j na alternativa k,

f_{jk} custo da estação j na alternativa k.

Defina u_{jk} como uma variável de decisão 0-1 (condição (iii) não é satisfeita) tal que:

u_{jk} = 1, se a alternativa k é escolhida para a estação j,
0, caso contrário.

onde $\#\#\#_{k=1,..,nj} u_{jk}=1$.

Para cada estação j, a escolha de capacidade é representada pelo vetor ($u_{j,1}$, $u_{j,2}$, ..., $u_{j,nk}$) em que todos os elementos são nulos, exceto um. Desta maneira, temos que $m_j = \#\#\#_{k=1,..,nj} m_{jk} u_{jk}$ e $\#\#\#_j = \#\#\#_{k=1,..,nj} \#\#\#_{jk} u_{jk}$ e assim, (25) e (26) dependem de u_{jk}. Seja **u**={u_{jk}, j=1,..,n; k=1,..,n_j}. Similarmente à seção 5.1, Bitran e Tirupati supuseram que **ca** e **cs** sejam independentes de mudanças da capacidade na rede (veja a discussão na seção 5.1). Como consequência, podemos primeiro resolver o sistema (25) para um dado **u** (i.e., uma dada capacidade **m** e ###) e então, tratar os **ca** e **cs** obtidos como parâmetros fixos em (26). Além disso, ao escolher a alternativa k na estação j (i.e., $u_{jk}=1$ e $u_{jl}=0$, *l*###k), podemos nos referir a $L_j(m_j, \#\#\#_j, ca_j, \#\#\#_j, cs_j)$ em (26) simplesmente como L_{jk}, onde $L_{jk}=L_j(m_{jk}, \#\#\#_j, ca_j, \#\#\#_{jk}, cs_j)$. Note que, desta maneira, podemos computar L_{jk} para toda alternativa k e toda estação j utilizando (26). Sem perda de generalidade, assumimos que se L_{jk}### L_{jl}, então f_{jk}###f_{jl}, k###*l*, k,*l*=1,..,n_j. Similarmente a (13), o WIP da rede pode ser reescrito como:

$$L(\mathbf{u}) = \#\#\#_{j=1,..,n} \ \#\#\#_{k=1,..,nj} \ v_j \ L_{jk}(u_{jk}) \quad (27)$$

onde v_j é o valor médio de um *job* na estação j, conforme anteriormente. Utilizando (27) obtemos o seguinte problema:

(SP1.1/G/M/R) minimizar $F(\mathbf{u}) = \#\#\#_{j=1,..,n} \#\#\#_{k=1,..,nj} f_{jk} u_{jk}$

sujeito a: $L(\mathbf{u}) \ \#\#\# \ L_T$

$\#\#\#_{k=1,..,nj} u_{jk} = 1, \quad j=1,..,n$

com: $u_{jk} \#\#\# \{0,1\}, \quad j=1,..,n, \ k=1,..,n_j$

onde L_T é uma dada meta para o WIP da rede. Note que supusemos L(**u**) e F(**u**) como funções lineares de **u**. SP1.1/G/M/R modela situações cujo alvo L_T é alcançado aumentando-se a capacidade por meio de máquinas adicionais, trabalhadores, ou aumentando-se a disponibilidade com horas extras e turnos adicionais de trabalho. BITRAN & TIRU-

PATI (1989b) propuseram um algoritmo heurístico (algoritmo 5) para resolver o programa linear inteiro em SP1.1/G/M/R acima. Eles mostraram que: (i) a solução ótima da relaxação LP de SP1.1/G/M/R tem *zero* ou *duas* diferentes variáveis u_{jk} com valores fracionários (proposição 3.1) e, (ii) se esta solução ótima tiver *duas* variáveis com valores fracionários, então elas correspondem à mesma estação (corolário). O algoritmo descrito abaixo produz a solução aproximada $\mathbf{u}^1$:

Algoritmo 5

1. Seja $\mathbf{u}^0$ a solução ótima da relaxação PL de SP1.1/G/M/R. Se $\mathbf{u}^0$ for uma solução factível de SP1.1/G/M/R, então $\mathbf{u}^1=\mathbf{u}^0$ é uma solução ótima de SP1.1/G/M/R e portanto, pare; caso contrário, vá para o passo 2.
2. Seja i a estação cujas variáveis são valores fracionários para certos k_1 e k_2 ($0 \# \# \# u_{i,k1}{}^0$, $u_{i,k2}{}^0 \# \# \# 1$). Uma solução factível de SP1.1/G/M/R é dada por:

$u_{jk}{}^1 = u_{jk}{}^0$, j###i, j=1,..,n, k=1,..,n_j,

$u_{ik}{}^1 = 1$, se k=*l*,

0, caso contrário,

onde *l* é tal que $L_{il} = \max\{L_{ik} \mid L_{ik} \# \# \# L_{i,k1}u_{i,k1}{}^0 + L_{i,k2}u_{i,k2}{}^0, k=1,..,n_i\}$.

Bitran e Tirupati também apresentaram um limitante para o erro do valor da solução aproximada $\mathbf{u}^1$ gerada pelo algoritmo 5. Sem perda de generalidade, assuma que $L_{i,k1}>L_{i,k2}$, e denote por $\mathbf{u}^*$ a solução ótima de SP1.1/G/M/R. Então,:

$$0 \# \# \# F(\mathbf{u}^1) - F(\mathbf{u}^*) \# \# \# f_{i,k1} - f_{i,k2} \# \# \# \max\{f_{jk}, j=1,..,n, k=1,..,n_j\}$$

Experiências computacionais com um exemplo de uma rede real com 13 estações e 10 classes de produtos indicaram que o algoritmo 5 pode ser uma boa aproximação para SP1.1/G/M/R quando o número de classes é relativamente grande. Neste exemplo, ao reduzir-se o WIP da rede de um valor inicial de 80000 para uma valor final abaixo de 30000, o erro relativo entre $F(\mathbf{u}^1)$ e $F(\mathbf{u}^*)$ foi menor do que 0.08%. A maior variação no valor de **ca** foi de 4.6%, correspondendo a uma variação de 0.5% no WIP (lembre-se de que os valores de **ca** e **cs** foram mantidos constantes no algoritmo). Uma perspectiva a ser ainda explorada é desenvolver abordagens para situações envolvendo um pequeno número de classes e misturas de roteiros determinísitcos e probabilísticos.

Um estudo utilizando refinamentos dos algoritmos 3, 4 e 5 para gerar curvas de *tradeoff* entre a capacidade e o nível de WIP num sistema *job-shop* pode ser encontrada em BITRAN & MORABITO (1995a). Naquele trabalho os autores ilustraram a forma de utilização dos algoritmos para avaliar a eficiência do sistema, decidir quanto de capacidade adquirir, como alocar recursos entre a redução de incerteza e introdução de novas tecnologias, e como avaliar o impacto de mudanças na taxa de produção e no mix de produtos.

5.3 Modelos SP1.1/G/M/N e SP2.1/G/M/N

VAN VLIET & RINNOOY KAN (1991) apresentaram dois algoritmos para resolver os modelos SP1.1/G/M/N e SP2.1/G/M/N, baseados em análise marginal e no método guloso. Estes algoritmos estão estreitamente relacionados com os dois algoritmos apresentados por BOXMA et al (1990) para resolver os modelos SP1.1/J/M/N e SP2.1/J/M/N (descritos na seção 4.2). Novamente, os *jobs* pertencem a múltiplas classes e cada classe percorre um roteiro determinístico diferente. Ao contrário da seção 5.2, as variáveis de decisão correspondem ao número de máquinas em cada estação.

Consideremos novamente o sistema linear em (25), e a expressão (26) para o número médio de *jobs* numa fila GI/G/m_j da estação j. Van Vliet e Rinnooy Kan consideraram cada capacidade m_j, j=1,...,n, como uma variável de decisão *inteira*. Dados ### e ###, (25) e (26) sugerem que mudanças na capacidade **m** resultam em mudanças nos **ca** e **cs** (L_j é função de m_1, m_2, ..., m_n). Note, entretanto, que esta relação funcional não é fácil de ser analisada.

Baseados nos resultados de BITRAN & TIRUPATI (1989a) (veja seção 5.1), Van Vliet e Rinnooy Kan assumiram que **ca** e **cs** são independentes de mudanças na capacidade **m**. Portanto, L_j não é dependente de m_i, i###j (condição (ii) satisfeita). Eles argumentaram que ao modificarmos **m**, a média e a variância do tempo de serviço variam na mesma proporção e assim, **cs** permanece aproximadamente constante. Além disso, a sensibilidade de **ca** com relação a mudanças em **m** parece ser pequena à medida que o número de classes aumenta e a proporção da carga devida a cada classe decresce. Portanto, uma vez que o conjunto de equações (25) tenha sido calculado, podemos ver **ca** e **cs** apenas como parâmetros em (26). Isto significa que $L_j(m_j, ###_j, ca_j, ###_j, cs_j)$ em (26) pode ser visto como uma função apenas de m_j, agora denotada por $L_j(m_j)$. Dado que $Lq_j(M/M/m_j)$ é uma função convexa em m_j e que estamos admitindo **ca** e **cs** como parâmetros, temos que $L_j(m_j)$ é uma função convexa em m_j (condição (i) satisfeita). Seguindo os mesmos passos da seção 4.2, o WIP da rede L(**m**) é definido conforme (3) (onde $L_j(m_j)$ é dado por (26), ao invés de (2)), e o número inicial de máquinas $\mathbf{m}^0$ deve satisfazer (4).

Modelo SP1.1/G/M/N

O modelo SP1.1/G/M/N é formulado exatamente como SP1.1/J/M/N descrito na seção 4.2, onde F(**m**) é uma função convexa não-decrescente de **m** (condição (iv) satisfeita), e L(**m**) é suposta convexa em **m**, conforme a discussão acima. SP1.1/G/M/N, também chamado problema de alocação de servidores, é um programa convexo com variáveis inteiras e portanto, o uso de análise marginal não leva necessariamente à otimalidade (condição (iii) não é satisfeita). O problema pode ser visto como um problema de alocação de máquinas a custo mínimo, tal que o WIP da rede resulte menor do que um dado valor-meta.

Van Vliet e Rinnooy Kan utilizaram o algoritmo 1 (seção 4.2) para resolver SP1.1/G/M/N. O algoritmo inicia com a menor alocação possível de máquinas $\mathbf{m}^0$ para todas as estações (alocação infactível). Em cada iteração, ele adiciona uma máquina na estação com o menor índice de prioridade (i.e., o quociente entre o aumento da função objetivo e a redução do WIP da rede). Note que este índice de prioridade é o resultado de análise marginal. Ele é obtido ao substituir (26) em (5). O algoritmo termina assim que a adição de uma máquina numa estação tornar a alocação factível.

O limitante de erro fornecido por BOX-MA et al (1990) (discutido na seção 4.2) também pode ser aqui aplicado. Por exemplo, se p é a última iteração do algoritmo 1 e **m*** é a solução ótima de SP1.1/G/M/N, temos que: $F(\mathbf{m}^{p-1}) < F(\mathbf{m}^*) ### F(\mathbf{m}^p)$. Van Vliet e Rinnooy Kan geraram curvas de *tradeoff* entre o custo e o WIP da rede, similares àquelas discutidas por BITRAN & TIRUPATI (1989a). A partir dos resultados computacionais de dois exemplos de OQN, eles encontraram um erro relativo de 7% na solução do primeiro exemplo, e de 5% na solução do segundo. Este erro relativo diminui à medida que o valor-meta escolhido para o WIP decresce. Van Vliet e Rinnooy Kan também recalcularam **ca** para a configuração final da rede, para verificar a sensibilidade de **ca** a

mudanças de **m**. Eles obtiveram um erro de **ca** abaixo de 6%, sugerindo que esta abordagem é uma boa aproximação para o problema SP1.1/G/M/N.

Modelo SP2.1/G/M/N

Similarmente, o modelo SP2.1/G/M/N é formulado exatamente como o modelo SP2.1/J/M/N descrito na seção 4.2, onde M é o número de máquinas disponíveis tais que $M>\#\#\#_{j=1,..,n} m_j^0$. Novamente, temos a função objetivo e a restrição do problema definidos como funções convexas de **m**. SP2.1/G/M/N, também chamado de problema de realocação de servidores, é um programa convexo com variáveis inteiras e pode ser visto como um problema de redistribuir M máquinas ao longo da rede de maneira a minimizar seu WIP.

Van Vliet e Rinnooy Kan utilizaram o algoritmo 2 (seção 4.2) para resolver SP2.1/G/M/N. O algoritmo inicia com a menor alocação possível de máquinas $\mathbf{m}^0$. Em cada iteração, ele adiciona uma máquina na estação com o máximo índice de prioridade (i.e., a maior redução de WIP por máquina). Note que este índice de prioridade é obtido por meio de análise marginal ao substituir (26) em (8). O algoritmo termina quando todas as M máquinas tiverem sido alocadas.

Uma vez que estamos assumindo que L(**m**) é uma função convexa em **m**, o algoritmo 2 termina após O(Mn) passos com a realocação ótima de máquinas (veja seção 4.2 para maiores detalhes). Experiências computacionais com os dois exemplos anteriores indicaram que a sensibilidade de **ca** a mudanças de **m** é pequena. Assim, a solução ótima produzida pelo algoritmo 2 para o suposto problema convexo pode ser utilizada como uma boa aproximação para o problema original.

6. Perspectivas para Futuras Pesquisas

Vários autores têm observado que os sistemas de manufatura modernos estão-se tornando muito complexos, devido principalmente a: (i) grande número de classes de produtos competindo pelos mesmos recursos, (ii) incerteza na demanda dos produtos, e (iii) redução dos ciclos de vida. Além de desenvolver métodos mais eficazes para analisar sistemas cada vez mais complexos, também podemos tentar re-duzir a complexidade no ambiente de manu-fatura. Grande parte do êxito de JIT e outros métodos relacionados é devida à simplificação. Exemplos de alternativas para reduzir a complexidade incluem a partição de linhas de produção existentes, a duplicação de recursos, e o reprojeto de produtos e proces-sos de manufatura. Note que os problemas da classe SP3 (p.e., o problema SP3.1 na seção 1), podem ser vistos neste contexto.

Recentes tentativas baseadas em modelos de OQN têm sido feitas para analisar os *tradeoffs* entre a partição de linhas de produção e a duplicação de máquinas (BITRAN & SARKAR, 1994c; veja também TANG & YOO, 1991, para um estudo relacionado de partição de clientes e alocação de servidores em um sistema de serviços com fila única). A idéia é relacionar os conceitos de *complexidade* e *previsibilidade* de um sistema, sugerindo que sistemas mais complexos tendem a ser menos previsíveis. Por exemplo, um gerente de produção deve ser capaz de prever o *leadtime* de um produto com razoável precisão. À medida que aumentamos o número de classes de produtos manufaturadas no sistema, a complexidade tende a crescer e a previsibilidade tende a diminuir devido à interferência entre as classes nas estações. Podemos também reduzir a complexidade mediante a *partição da planta*. A seguir, sugerimos possíveis

medidas de desempenho que capturam esta noção. Consideramos:

(i) medidas de complexidade do ponto de vista da gerência de produto,

(ii) medidas de complexidade do ponto de vista da gerência de estação.

6.1 Medidas de Complexidade do Ponto de Vista da Gerência de Produto

Gerentes devem ser capazes de prever o *leadtime* de um produto o mais precisamente possível. Em outras palavras, a variância do *leadtime* deve ser pequena. Podemos reduzir a variância adicionando máquinas nas estações. Seja T_k o *leadtime* de um produto da classe k, w_k um peso associado ao produto da classe k, e T um limitante superior para o *leadtime* ponderado de todas as classes na rede. Assim, podemos formular a seguinte restrição de complexidade:

$$\#\#\#_{k=1,r}\ w_k\ V(T_k)\ \#\#\#\ T \qquad (28)$$

Note que quanto menor for o valor fixado de T, maior é a previsibilidade no sistema. Cada variância $V(T_k)$ em (28) é definida como a soma das variâncias dos tempos de espera nas filas e dos tempos de serviço de todas as estações no roteiro da classe k. Por simplicidade, vamos supor que os tempos de serviço sejam determinísticos em todas as estações e portanto, suas variâncias são nulas. Obtemos então (BITRAN & SARKAR, 1994c):

$$V(T_k) = \#\#\#_{l=1,..,nk}\ \#\#\#_{j=1,..,n}\ V(Wq_j)\ 1\{j:\ j=n_{kl}\} \qquad (29)$$

onde $V(Wq_j)$ é a variância do tempo de espera na estação j, dada por:

$$V(Wq_j) = [Wq_j(M/M/m_j)]^2\ (ca_j + cs_j) / 4 \qquad (30)$$

onde $Wq_j(M/M/m_j)$ é o tempo de espera médio num sistema $M/M/m_j$. Uma vez que $Wq_j(M/M/m_j)$ é uma função convexa decrescente em m_j e assumindo-se que ca_j e cs_j sejam independentes de mudanças de capacidade na rede (veja seção 5.1 para uma discussão detalhada), segue de (30) que $V(Wq_j)$ decresce ao aumentarmos m_j. Portanto, ao adicionarmos máquinas nas estações do roteiro da classe k, reduzimos a variância $V(T_k)$ em (29) e assim, a complexidade do sistema do lado esquerdo de (28).

As expressões (28)-(30) também sugerem que, para a mesma capacidade total da planta, poderíamos reduzir a complexidade do sistema ao particionarmos apropriadamente a planta em sub-plantas (ou linhas de produção) com um mix de produtos mais homogêneo. Desta maneira, obteríamos parâmetros de variabilidade menores nas estações de cada sub-planta, tais que a variância total de cada classe em (29) fosse reduzida, e por conseguinte, a complexidade do sistema em (28).

6.2 Medidas de Complexidade do Ponto de Vista da Gerência de Estação

BITRAN & SARKAR (1994c) observaram que à medida que o número de máquinas cresce em uma estação, esperamos obter maior *flexibilidade* (em termos de programação e manutenção) para operar essa estação. Para determinar o número de máquinas em cada estação, devemos considerar as incertezas dos intervalos de tempo entre-chegadas e dos tempos de processamento dos produtos que visitem a estação.

WHITT (1992) discutiu algumas heurísticas que podem ser úteis para descrever restrições de complexidade com respeito ao *nível de serviço* de cada estação. O nível de serviço é uma medida que, quando fixada, mantém uma certa *medida de congestão* aproximadamente constante na estação (logo abaixo discutiremos a relação entre o nível de serviço e a flexibilidade da estação). Por

exemplo, seja $###_j$ o nível de serviço na estação j, dado por:

$$###_j = (1 - ###_j)\ ###m_j \qquad (31)$$

A expressão (31) sugere uma *economia de escala*, isto é, para um certo nível de serviço, a utilização média aumenta ao aumentarmos o número de máquinas e a taxa média de chegada $###_j$ da estação j (lembre-se de que $###_j=###_j/(m_j###_j)$). Note, entretanto, que a taxa de crescimento do número de máquinas é menor do que a taxa de crescimento da taxa média de chegada. Whitt mostrou que se mantivermos $###_j$ constante em (31), também estamos mantendo a medida de congestão $P(W_j>0)$ aproximadamente constante (i.e., a probabilidade de um tempo de espera positivo). Este resultado é suportado por teoremas do limite de tráfego-pesado, e foi observado em experimentos computacionais. Em particular, Whitt mostrou que para uma fila $GI/G/m_j$, temos a seguinte aproximação:

$$###_j\ ###\ (ca_j + cs_j) / 2\ ###m_j\ E(W_j|W_j\ ###0) \qquad (32)$$

onde $E(W_j|W_j###0)$ também é uma medida de congestão. Ela corresponde ao tempo médio de espera na fila da estação j, dado que o tempo de espera é maior do que zero. Combinando (31) e (32), obtemos um outro exemplo de nível de serviço para a estação j definido como:

$$###_j = (1 - ###_j)\ m_j / (ca_j + cs_j)\ ###\ 1 / 2\ E(W_j|W_j###0) \qquad (33)$$

A equação (33) implica que para um dado nível de serviço $###_j$, mantemos a medida de congestão $E(W_j|W_j###0)$ aproximadamente constante. Assumindo-se que ca_j e cs_j sejam independentes de mudanças de capacidade na rede (veja seção 5.1 para uma discussão detalhada), segue de (33) que ao adicionarmos máquinas e aumentarmos a taxa média de chegada na estação j, a utilização média cresce para o mesmo nível de serviço. Seja a seguinte restrição para cada estação j da rede:

$$(1 - ###_j)\ m_j / (ca_j + cs_j)\ ###\ G_j \qquad (34)$$

onde G_j é um limitante inferior para o nível de serviço na estação j (note que G_j também é um limitante superior para a medida de congestão $E(W_j|W_j>0)$). O parâmetro G_j pode ser visto como a mínima flexibilidade desejada na estação j. Para maiores valores de ca_j e cs_j, devemos aumentar o número de máquinas na estação j de maneira a satisfazer a flexibilidade desejada.

A expressão (34) sugere que, em certos casos, podemos satisfazer a mínima flexibilidade desejada nas estações sem alterar o número total de máquinas na rede. Ao particionarmos apropriadamente a planta em sub-plantas com um mix de produtos mais homogêneo, podemos obter parâmetros de variabilidade menores, digamos ca_j^i e cs_j^i para cada estação j de cada sub-planta i, tal que o lado esquerdo de (34) cresça para todo j e i.

6.3 Projeto de Fábrica Focalizada

O problema do *projeto de fábrica focalizada* envolve a alocação de produtos em linhas de produção, e a alocação de capacidade em estações de cada linha. Este problema pode ser visto como um exemplo da classe SP3 (veja, p.e., o problema SP3.1 na seção 1), e é diferente dos problemas discutidos na seções 4 e 5 em que apenas a alocação de capacidade estava envolvida.

Um tópico de pesquisa interessante é desenvolver modelos de otimização para analisar o *tradeoff* entre a partição de linhas

de produtos e a duplicação de máquinas num projeto de fábrica focalizada. Poderíamos incorporar nestes modelos restrições de complexidade dos pontos de vista do produto e da estação, tais como (28) e (34) discutidas acima. Estas restrições podem ajudar-nos a representar os *leadtimes* desejados de produtos e as flexibilidades desejadas nas estações no sistema.

Uma pesquisa recente baseada nestas idéias (BITRAN & SARKAR, 1994c) revela um resultado inesperado:

"Ao contrário do senso comum, o número de máquinas requerido pode ser menor quando as linhas de produção são particionadas (comparado a quando elas são colocadas em uma única planta)."

Este resultado sugere que podemos reduzir a complexidade da rede (lado esquerdo de (28)), ou aumentar a flexibilidade das estações (lado esquerdo de (34)), apenas particionando a planta em linhas de produto. Pesquisa adicional poderia investigar a estabilidade das partições ótimas. Por exemplo, a sensibilidade da solução a mudanças na taxa média de chegada de produtos, ou a mudanças no nível de serviço desejado nas estações. Poderíamos também considerar situações particulares, em que classes de produtos privilegiados deveriam ter baixos *leadtimes*, ou percorrer roteiros através de estações com altos níveis de serviço.

Agradecimentos

Os autores gostariam de agradecer ao Dr. Deb Sarkar da AT&T Bell Laboratories, a Luis A. C. Pedrosa da Sloan School of Management, MIT, e aos três *referees* pelos seus úteis comentários e sugestões. Esta pesquisa contou com o apoio da FAPESP, mediante concessão de uma bolsa de pós-doutorado, processo #93/0891-7.

Referências Bibliográficas:

ALBIN, S. L.: "Delays for customers from different arrival streams to a queue". *Mgmt. Sci.* 32, 329-340, 1986.

BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M.: *Nonlinear programming: Theory and algorithms*, 2nd.ed., Wiley, NY, 1993.

BITRAN, G. R. & DASU, S.: "A review of open queueing network models of manufacturing systems". *Queueing Syst.* 12, 95-134, 1992.

BITRAN, G. R. & MORABITO, R.: "Open queueing networks: Optimization and performance evaluation models for discrete manufacturing systems". *WP#3743-94, Sloan School of Management, MIT,* 45p, 1994 (aceito para publicação no *Production and Oper. Mgmt.*, 1995).

BITRAN, G. R. & MORABITO, R.: "Manufacturing systems design: *Tradeoff* curve analysis". *WP#3805-95, Sloan School of Management, MIT,* 33p, 1995a.

BITRAN, G. R. & MORABITO, R.: "Um exame dos modelos de redes de filas abertas aplicados a sistemas de manufatura discretos---Parte I". *Gestão & Produção* 2(2), Agosto 1995, 109-220, 1995b.

BITRAN, G. R. & SARKAR, D.: "Throughput analysis in manufacturing networks". *EJOR* 74, 448-465, 1994a.

BITRAN, G. R. & SARKAR, D.: "Targeting problems in manufacturing queueing networks - An iterative scheme and convergence". *EJOR* 76, 501-510, 1994b.

BITRAN, G. R. & SARKAR, D.: "Focused factory design: Complexity, capacity and inventory *tradeoffs*". *Technical Memorandum, AT&T Bell Lab.*, 36p, 1994c.

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

BITRAN, G. R. & TIRUPATI, D.: "*Tradeoff* curves, targeting and balancing in manufacturing queueing networks". *Oper. Res.* 37, 547-564, 1989a.

BITRAN, G. R. & TIRUPATI, D.: "Capacity planning in manufacturing networks with discrete options". *Annals of Oper. Res.* 17, 119-136, 1989b.

BOXMA, O. J.; RINNOOY KAN, A.; VAN VLIET, M.: "Machine allocation problems in manufacturing networks". *EJOR* 45, 47-54, 1990.

BUZACOTT, J. A. & SHATHIKUMAR, J. G.: "Design of manufacturing systems using queueing models". *Queueing Syst.* 12, 135-214, 1992.

BUZACOTT, J. A. & SHANTHIKUMAR, J. G.: *Stochastic models of manufacturing systems*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

BUZACOTT, J. A. & YAO, D. D.: "Flexible manufacturing systems: A review of analytical models". *Mgmt. Sci.* 32(7), 890-905, 1986.

CALABRESE, J. M.: "Optimal workload allocation in open networks of multiserver queues". *Mgmt. Sci.* 38(12), 1792-1802, 1992.

CHASE, R. B. & AQUILANO, N. J.: *Production and operations management - A life cycle approach*, Irwin, Homewood, MA, 1992.

DALLERY, Y. & STECKE, K. E.: "On the optimal allocation of servers and workloads in closed queueing networks". *Oper. Res.* 38(4), 694-703, 1990.

DISNEY, R. L. & KONIG, D.: "Queueing networks: A survey of their random processes". *SIAM Rev.* 27(3), 335-403, 1985.

ERLANG, A. K.: "Solution of some problems in the theory of probabilities of some significance in automatic telephone exchanges". *Post Office Electrical Engineer's Journal* 10, 189-197, 1917.

HAREL, A. & ZIPKIN, P.: "Strong convexity results for queueing systems". *Oper. Res.* 35, 405-418, 1987.

TANG, C. S. & YOO, S.: "System planning and configuration problems for optimal system design". *EJOR* 54, 163-175, 1991.

HARRISON, J. & NGUYEN, V.: "Brownian models of multiclass queueing networks: Current status and open problems". *Queueing Syst.* 13, 5-40, 1993.

HARRISON, J. & WILLIAMS, R.: "Brownian models of open queueing
networks with homogeneous customer populations". *Stochastic* 22, 77-115, 1987.

HSU, L. F.; TAPIERO, C. S.; LIN, C.: "Network of queues modeling in flexible manufacturing systems: A survey". *RAIRO* 27(2), 201-248, 1993.

KELLY, F. P.: *Reversibility and Stochastic Processes*, Wiley, NY, 1979.

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

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

KOENIGSBERG, E.: "Twenty five years of cyclic queues and closed queue networks: A review". *J. Opl. Res. Soc.* 33, 605-619, 1982.

LEMOINE, A. J.: "Networks of queues - a survey of equilibrium analysis". *Mgmt. Sci.* 24, 464-481, 1977.

REIMAN, M. I.: "Open queueing networks in heavy-traffic". *Math. of Oper. Res.* 9, 441-458, 1984.

SHANTHIKUMAR, J. G. & YAO, D. D.: "Optimal server allocation in a system of multi-server stations". *Mgmt. Sci.* 33(9), 1173-1180, 1987.

SHANTHIKUMAR, J. G. & YAO, D. D.: "On server allocation in multiple center manufacturing systems". *Oper. Res.* 36(2), 333-342, 1988.

SUNDARRAJ, R. P.; SUNDARARAGHAVAN, P. S.; FOX, D. R.: "Optimal server acquisition in open queueing networks". *J. Oper. Res. Soc.* 45(5), 549-558, 1994.

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

VAN VLIET, M. & RINNOOY KAN, A.: "Machine allocation algorithms for *job shop* manufacturing". *Journal of Intelligent Manufacturing* 2, 83-94, 1991.

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

WEIN, L. M.: "Scheduling networks of queues: Heavy traffic analysis of a two-station network with controllable inputs". *Oper. Res.* 38(6), 1065-1078, 1990b.

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

WHITT, W.: "Understating the efficiency of multi-server service systems". *Mgmt. Sci.* 38(5), 708-723, 1992.

WHITT, W.: "Approximations for the GI/G/m queue". *Production and Oper. Mgmt.* 2(2), 114-161, 1993.

A SURVEY ON OPEN QUEUEING NETWORK MODELS APPLIED TO DISCRETE MANUFACTURING SYSTEMS: PART II

Abstract

This paper presents the second (and last) part of our survey on open queueing network models applied to discrete manufacturing systems. We focus on design and planning for job-shops. In the first part (Bitran and Morabito, 1995b) we reviewed exact and approximate decomposition methods for performance evaluation models for single and multiple product class systems. The second part reviews optimization models of three categories of problems: the first minimizes capital investment subject to attaining a performance measure (WIP or leadtime), the second seeks to optimize the performance measure subject to resource constraints, and the third explores recent research developments in complexity reduction through shop redesign and products partitioning.

Key-words: ***open queueing networks, optimization and performance evaluation, decomposition methods, discrete manufacturing systems, job-shop design.***