Programação da produção em sistema *no-wait flow shop* com minimização do tempo total de fluxo

Recebido em: 09/08 Avaliado em: 11/09

Lucas Yamada Scardoelli (EESC/USP) scarty@terra.com.br
• R. General Glicério, 340, Centro, CEP 15900-000, Taquaritinga, SP
Marcelo Seido Nagano (EESC/USP) – drnagano@usp.br
João Vitor Moccellin (EESC/USP) – jvmoccel@sc.usp.br

Resumo
Este artigo trata do problema de programação de operações em um ambiente de produção flow shop*, sem interrupção na execução das tarefas, tendo como objetivo a minimização do estoque em processamento. Dois novos métodos heurísticos são propostos para a solução do problema. Por meio de uma experimentação computacional, os desempenhos dos novos métodos são avaliados e comparados com o melhor método heurístico reportado na literatura. Os resultados experimentais mostram a superioridade dos novos métodos para o conjunto de problemas avaliados.*
Palavras-chave: *estoque de processo; no-wait* flow shop*; métodos heurísticos.*

Abstract
This paper deals with the no-wait flow shop scheduling problem with the objective of minimizing total flow time, therefore reducing in-process inventory. Two news heuristic methods are proposed for the scheduling problem solution. The performance of the news methods are evaluated and compared with the best heuristic reported in the literature. Experimental results show the new methods superiority for the test problems set.
Keywords: *in-process inventory, no-wait flow shop, heuristics methods.*

1. INTRODUÇÃO

O problema tradicional de programação *flow shop* é um problema de produção, onde um conjunto de *n* tarefas deve ser processado, na mesma seqüência, por um conjunto de *m* máquinas. Quando a ordem de processamento em todas as máquinas for a mesma, tem-se o ambiente de produção *Flow shop* Permutacional, onde o número de possíveis programações para *n* tarefas é *n*!.

Uma situação específica para este problema, mas muito freqüente, é a programação da produção em sistema *flow shop*, sem interrupção na execução das tarefas, que geralmente ocorre em um ambiente caracterizado pela tecnologia do processo, i.é., quando, por exemplo, uma variação de temperatura pode influenciar na degeneração do material ou pela falta de capacidade de armazenamento entre as máquinas.

Este problema também, pode ser chamado de *flow shop* sem espera ou *no-wait flow shop* (*NWFS*), conforme apresentado na figura 1.

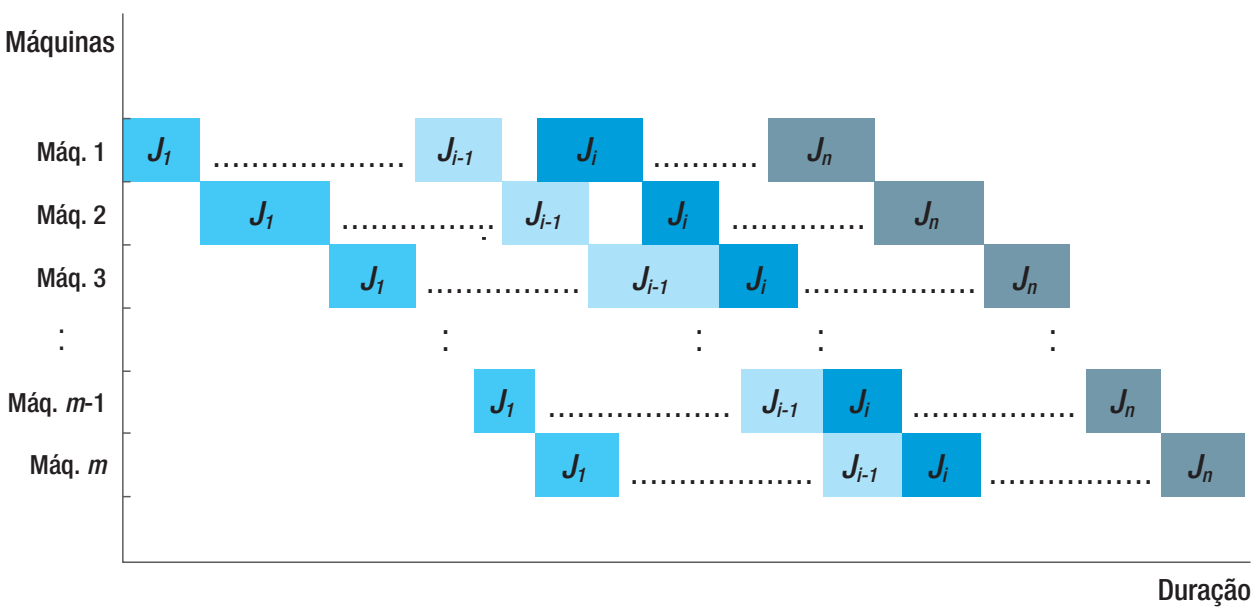

FIGURA 1 – *No-Wait Flow shop* com m máquinas e n tarefas.

A figura 1 apresenta a programação de um conjunto de *n* tarefas ($J = \{J_1,J_2,...,J_i,...,J_n\}$) que devem ser processadas, na mesma seqüência, por um conjunto de *m* máquinas distintas, onde o tempo de processamento da tarefa J_i na máquina *k*, é t_{ik} ($i = 1,2,...,n$; $k = 1,2,...,m$).

A principal característica do *NWFS* é a necessidade de que a operação $g + 1$ de uma determinada tarefa J_i tem que ser processada logo após o término da operação g ($1 \leq g \leq m - 1$), não permitindo que ocorra tempo de espera no processamento de uma tarefa de uma máquina para a outra (*no-wait*). O único tempo de espera permitido é o início do processamento da primeira operação das tarefas, na seqüência, na primeira máquina. Devido à sua natureza, o problema *NWFS* é considerado como NP-hard, para o caso de três ou mais máquinas (CHEN et al., 1996).

O objetivo deste trabalho é apresentar dois novos métodos heurísticos, para o problema de programação de operações em ambientes *NWFS*, com o critério de minimização do tempo total de fluxo das tarefas, verificando o desempenho, com o melhor método heurístico reportado na literatura.

As características essenciais a serem consideradas para os novos métodos, são: equilíbrio entre a qualidade da solução e eficiência computacional, simplicidade e facilidade de implementação.

2. PROBLEMA DE PROGRAMAÇÃO DE OPERAÇÕES *NWFS*

Deman e Baker (1974) foram os primeiros a estudar o problema *NWFS*, com o critério de minimização do tempo total de fluxo das tarefas. Em sua pesquisa, eles apresentaram um algoritmo branch & bound para estabelecer todas as seqüências parciais, utilizando-se limitantes inferiores. Os autores concluíram que os resultados obtidos foram satisfatórios e que o problema podia ser solucionado tão rapidamente quanto os problemas que utilizavam, como critério, a minimização da duração total da programação (makespan).

Rajendran e Chaudhuri (1990) apresentaram dois algoritmos heurísticos, constituídos de duas fases: uma primeira fase de ordenação inicial das tarefas e uma outra, de construção da seqüência final, com avaliação de seqüências parciais. A experimentação computacional revelou que as soluções obtidas pelos novos métodos foram melhores, quando comparadas com os métodos anteriormente, propostos por Bonney e Gundry (1976) e King e Spachis (1980).

Chen et al. (1996) desenvolveram uma meta heurística, baseada no algoritmo genético, e através da sua parametrização, obtiveram melhores soluções que o melhor método proposto por Rajendran e Chaudhuri (1990).

Bertolissi (1999) apresentou um método heurístico de apenas uma fase, na qual é definida uma parcela da soma dos tempos de fluxo de duas tarefas adjacentes J_i e J_j, a partir do início de J_i (ou seja, no intervalo de tempo entre o início de J_i e o termino de J_j); e que, em seguida, é obtido uma ordenação das tarefas. Os resultados experimentais mostraram que o método proposto não obtém soluções melhores que os métodos de Rajendran e Chaudhuri (1990).

Bertolissi (2000) apresentou um novo método heurístico, composto de duas fases. Na primeira fase, similar a Bertolissi (1999), em que é definida uma ordenação inicial das tarefas, de acordo com as parcelas das somas dos tempos de fluxo dos pares tarefas adjacentes J_i e J_j. Na segunda fase, é utilizado o procedimento similar de inserção de tarefas de Rajendran e Chaudhuri (1990). Os resultados experimentais permitiram concluir que o método heurístico proposto obtém soluções melhores, quando comparado aos métodos heurísticos de Rajendran e Chaudhuri (1990).

Fink e Voβ (2003) utilizaram os procedimentos meta heurísticos de Busca Tabu e Simulated Anneling e concluíram que a efetividade dos procedimentos meta heurísticos utilizados na solução do problema, dependem da qualidade da solução desejada e do tempo computacional disponível. Ou seja, os autores não contemplam um método como o melhor entre todos, mas afirmam que, dependendo das características do problema, existe um que pode ser mais adequado.

Aldowaisan e Allahverdi (2004) propuseram novos métodos heurísticos, compostos de três fases. Segundo os resultados dos experimentos computacionais, o algoritmo PH1(p) foi o que apresentou os melhores resultados, comparados com os métodos de Rajendran e Chaudhuri (1990) e o método metaheurístico de Chen et al. (1996). O método proposto é composto pelas seguintes fases: na primeira fase, é desenvolvida uma ordenação inicial, a partir da tarefa que apresenta a menor soma dos tempos de processamento e, em seguida, são ordenadas as tarefas que resultam no menor tempo total de fluxo na ordenação parcial. Na segunda fase, é obtida uma solução inicial semelhante ao método de inserção, proposto por Nawaz et al. (1983). Na última fase, é aplicado o procedimento de melhoria da solução atual utilizando-se a vizinhança de permutação de pares de tarefas.

3. MÉTODOS HEURÍSTICOS PROPOSTOS

Os dois métodos heurísticos propostos neste trabalho possuem uma única fase e serão denominados SN-1 e SN-2, para fins de comparação com o melhor método da literatura.

Os métodos heurísticos propostos têm como base a expressão apresentada por Bertolissi (2000), que trata de uma parcela da soma dos tempos de fluxo de duas tarefas adjacentes J_i e J_j, a partir do início de J_i (ou seja, no intervalo de tempo entre o início de J_i e término de J_j) para m máquinas.

O cálculo das parcelas das somas dos tempos de fluxo, para o conjunto de pares de tarefas adjacentes para m máquinas, pode ser obtida pela seguinte expressão:

$$F_m(J_i,J_j) = 2t_{i1} + \sum_{k=2}^{m} t_k + R_m(J_i,J_j) \quad (1)$$

$i = 1,2...,n,\ k = 1,2...,m$

Onde:

$$R_1(J_i,J_j) = t_{i1}\ ,\ R_1(J_i,J_j) = t_{jm} + max\left(R_{m-1}(J_i,J_j), \sum_{k=2}^{m} t_{ik}\right)$$

Na expectativa de minimizar o tempo total de fluxo da tarefas, a expressão 1 possibilita ordenar as tarefas, a partir das menores parcelas das somas dos tempos de fluxo de pares de tarefas adjacentes para m máquinas.

A seguir, serão apresentados os algoritmos dos métodos SN-1 e SN-2, para solução do problema *NWFS*, considerado neste trabalho.

1.1. Método Heurístico SN-1

Seja, $J = \{J_1,J_2,...,J_i,...,J_n\}$ o conjunto de n tarefas, que devem ser programadas e S o conjunto da seqüência das tarefas programadas; S_p o conjunto da melhor seqüência obtida, utilizando-se a vizinhança de permutação de pares de tarefas; e S_i o conjunto da melhor seqüência obtida, utilizando-se a vizinhança de inserção de tarefa.

O método heurístico SN-1 é composto pelos seguintes passos:

Passo 1

$J = \{J_1,J_2,...,J_i,...,J_n\}$;

$S = \varnothing$;

Selecione o menor elemento $F_m(J_i,J_j)$;

$S\ \{J_i,J_j\}$;

$J \leftarrow J - \{J_i,J_j\}$;

$u \leftarrow J_{[2]}$;

$k = 3$;

Passo 2

Selecione o menor elemento $F_m(J_u,J_v)$ tal que $J_v \in J$;

$S \leftarrow S \cup \{J_v\}$;

$J \leftarrow J - \{J_v\}$;

$u \leftarrow J_{[k]}$;

$k \leftarrow k + 1$;

Passo 3

- Considerando toda a vizinhança de permutação da seqüência parcial com $(k-1)$ tarefas, constituída de $(k-1)(k-2)/2$ seqüências, determine a seqüência S_p, associada à menor soma dos tempos de fluxo;
- Considerando toda a vizinhança de inserção da seqüência parcial, com $(k-1)$ tarefas, constituída de $(k-2)^2$ seqüências, determine a seqüência S_i associada a menor soma dos tempos de fluxo;
- Se S_p é melhor que S e S_i, atualize S com S_p;
- Se S_i é melhor que S e S_p, atualize S com S_i;
- Volte ao Passo 2 até que todas as tarefas estejam seqüenciadas.

3.2. Método Heurístico SN-2

Passo 1

$J = \{J_1, J_2, ..., J_i, ..., J_n\}$;

$S = \varnothing$;

Selecione o menor elemento $F_m(J_i, J_j)$;

$S \{J_i, J_j\}$;

$J \leftarrow J - \{J_i, J_j\}$;

$u \leftarrow J_{[2]}$;

$k = 3$;

Passo 2

Selecione o menor elemento $F_m(J_u, J_v)$ tal que $J_v \in J$;

$S \leftarrow S \cup \{J_v\}$;

$J \leftarrow J - \{J_v\}$;

$u \leftarrow J_{[k]}$;

$k \leftarrow k + 1$;

Passo 3

- Considerando toda a vizinhança de inserção da seqüência parcial, com $(k-1)$ tarefas, constituída de $(k-2)^2$ seqüências, determine a seqüência S_i associada à menor soma dos tempos de fluxo;
- Se S_i é melhor que S, atualize S com S_i e, considerando toda a vizinhança de permutação da seqüência parcial com $(k-1)$ tarefas, constituída de $(k-1)(k-2)/2$ seqüências, determine a seqüência S_p associada à menor soma dos tempos de fluxo;
- Se S_p é melhor que S, atualize S com S_p;
- Se S_i não é melhor que S, considerando toda a vizinhança de permutação da seqüência parcial com $(k-1)$ tarefas, constituída de $(k-1)(k-2)/2$ seqüências, determine a seqüência S_p associada à menor soma dos tempos de fluxo;
- Se S_p é melhor que S, atualize S com S_p;
- Volte ao Passo 2, até que todas as tarefas estejam seqüenciadas.

A principal diferença entre os dois métodos decorre da combinação na aplicação dos procedimentos de melhoria, que utilizam as vizinhanças de permutação e inserção de tarefas. No método SN-1, os dois

procedimentos de melhoria são aplicados de forma simultânea na seqüência parcial *S*. Já no método SN-2, primeiro é aplicado o procedimento de melhoria que utiliza a vizinhança de inserção de tarefas, na seqüência parcial *S* e, em seguida, é aplicado o procedimento de melhoria que utiliza a vizinhança de permutação de tarefas, na seqüência parcial atualizada *S*.

Em seguida, serão apresentadas a experimentação computacional e a análise dos resultados do experimento.

4. EXPERIMENTAÇÃO COMPUTACIONAL

Os métodos heurísticos propostos (SN-1 e SN-2) foram comparados com o melhor método heurístico reportado na literatura, ou seja, o algoritmo heurístico, composto de três fases PH1(p), desenvolvido por Aldowaisan e Allahverdi (2004).

Na experimentação, foram avaliados 7.200 problemas, divididos em três grupos. O primeiro grupo de 2.000 problemas, considerados de pequeno porte, subdivididos em 20 classes, de acordo com o número de tarefas $n \in \{5,6,7,8,9\}$ e o número de máquinas $m \in \{5,10,15,20\}$. O segundo grupo, envolveu 3.200 problemas, considerados de médio porte, com número de tarefas $n \in \{10,20,30,40,50,60,70,80\}$ e máquinas $m \in \{5,10,15,20\}$, totalizando 32 classes. O terceiro grupo foi constituído por 2.000 problemas, considerados de grande porte, com número de tarefas $n \in \{90,100,110,120,130\}$ e $m \in \{5,10,15,20\}$, totalizando 20 classes.

Dessa forma, para cada classe (n,m), foram testados 100 problemas. Os tempos de processamento das operações foram gerados aleatoriamente, a partir de uma distribuição uniforme no intervalo $[1,9]$. Neste trabalho, os métodos foram codificados em linguagem Delphi e o equipamento utilizado foi um microcomputador Intel Celeron 2,66 GHz, com 256 MB de memória RAM e disco rígido de 40 Gb.

5. MÉTODO DE ANÁLISE

As estatísticas usadas para avaliar o desempenho dos métodos, foram a Porcentagem de Sucesso, o Desvio Médio Relativo e o Tempo Médio de Computação.

A Porcentagem de Sucesso é definida pelo quociente entre o número total de problemas, para os quais o método obteve a melhor solução (menor tempo total de fluxo das tarefas) e o número total de problemas resolvidos. Obviamente, quando dois ou mais métodos obtêm a melhor solução para o mesmo problema, suas Porcentagens de Sucesso são simultaneamente melhoradas.

O Desvio Relativo (DR_{hi}) quantifica o desvio que o método *h* obtém em relação a melhor solução para um mesmo problema *i*, sendo calculado pela expressão 2:

$$DR_{hi} = \left(\frac{F_{hi} - F_{*i}}{F_{*i}} \right) \qquad (2)$$

Onde:

F_{hi} = Tempo total de fluxo das tarefas obtido pelo método *h* para o problema *i*;
F_{*i} = Melhor tempo total de fluxo das tarefas obtido para o problema *i*.

O Tempo Médio de Computação (em segundos) é calculado pela soma dos tempos de processamento para uma determinada classe de problema (n,m), dividido pelo número total de problemas resolvidos.

Neste trabalho, todos os resultados apresentados foram agrupados com relação ao número de máquinas, possibilitando uma análise global dos resultados.

6. ANÁLISE DOS RESULTADOS

A tabela 1 apresenta os resultados obtidos dos métodos PH1(p), SN-1 e SN-2.

TABELA 1 – Resultados globais entre os algoritmos

Número de Tarefas	PH1(p)	SN-1	SN-2
5	88,25* **0,09**** 0,0006***	**93,25** 0,10 **0,0003**	91,00 0,11 0,0005
6	84,75 0,18 **0,0002**	87,25 0,11 0,0003	**90,50** **0,07** 0,0003
7	77,75 0,22 0,0006	79,25 0,19 **0,0002**	**83,00** **0,15** **0,0002**
8	65,25 0,37 0,0010	73,25 0,22 **0,0004**	**79,25** **0,21** 0,0005
9	60,25 0,41 0,0015	67,00 0,28 0,0006	**69,25** **0,26** **0,0005**
10	52,75 0,48 0,0012	58,75 0,39 0,0006	**64,50** **0,36** **0,0005**
20	26,25 0,95 0,0031	35,75 0,49 **0,0022**	**46,00** **0,45** 0,0023
30	20,00 1,05 **0,0084**	34,25 0,52 0,0086	**47,00** **0,39** 0,0092
40	17,25 1,33 **0,0162**	33,00 0,58 0,0223	**49,75** **0,37** 0,0218
50	14,00 1,35 **0,0291**	34,25 0,56 0,0492	**51,75** **0,32** 0,0497
60	15,75 1,43 **0,0475**	29,75 0,52 0,0977	**54,50** **0,26** 0,0978
70	11,50 1,56 **0,0722**	31,00 0,55 0,1770	**57,50** **0,28** 0,1774
80	10,25 1,58 **0,1061**	30,50 0,54 0,2970	**59,25** **0,22** 0,2971

Número de Tarefas	PH1(p)	SN-1	SN-2
90	7,50 1,66 **0,1453**	26,50 0,56 0,5030	**66,00** **0,18** 0,5131
100	5,25 1,71 **0,1945**	30,00 0,50 0,7114	**65,00** **0,17** 0,7125
110	4,75 1,89 **0,2568**	24,00 0,57 1,0924	**71,25** **0,15** 1,0871
120	3,75 1,95 **0,3485**	30,75 0,53 1,6407	**65,50** **0,17** 1,6418
130	4,00 1,91 **0,4392**	29,25 0,53 2,2438	**66,75** **0,13** 2,2443

*Porcentagem de Sucesso (%)..

**Desvio Médio Relativo (%)..

***Tempo Médio Computacional (segundo)..

Em uma primeira análise, verificou-se que a porcentagem de sucesso do método SN-2 foi superior, em comparação a todos os outros métodos no intervalo de variação de 6 a 130 tarefas.

Além disso, é possível concluir que para problemas com 20 a 90 tarefas, o método SN-2 apresentou tendência de aumento da porcentagem de sucesso, em relação a outros métodos.

Ainda com relação aos problemas de grande porte, verificou-se que o método PH1(p), apresentou porcentagem de sucesso inferior a 8%, em todos os casos, enquanto o método SN-2 apresentou porcentagem de sucesso superior de 65%.

Os métodos SN-1 e SN-2 apresentaram menor desvio relativo médio que o método PH1(p), apresentando soluções de melhor qualidade.

Verifica-se na tabela 1 que, para os problemas de grande porte, a heurística SN-2 também, foi a que apresentou menor porcentagem de desvio relativo médio (abaixo de 0,2%), enquanto o segundo melhor método apresentou uma porcentagem de desvio relativo médio acima de 0,5%.

De forma geral, entre todos os métodos avaliados nesta experimentação, verificou-se que o método SN-2 foi o que apresentou melhor desempenho, tanto em termos de porcentagem de sucesso quanto em porcentagem de desvio relativo médio.

Os métodos SN-1 e SN-2 apresentaram para os problemas de médio e grande porte, maiores tempos médios de computação, comparados ao PH1(p), mas para fins práticos esses valores podem ser considerados não relevantes.

7. CONSIDERAÇÕES FINAIS

Preliminarmente, ressalta-se que para fins práticos, as soluções obtidas pelos métodos heurísticos já existentes, para o problema *NWFS*, são suficientes, ou seja, tal problema pode ser considerado como já resolvido.

Tendo em vista, porém, a complexidade do problema em questão, a busca por novos métodos que contêm adequado equilíbrio entre a qualidade da solução e a eficiência computacional, simplicidade e facilidade de implementação, ainda continua como uma direção para novas pesquisas.

Dessa forma, este trabalho apresentou uma contribuição que procura evidenciar que apesar dos algoritmos existentes proporcionarem boas soluções, é possível, por meio de uma propriedade (BERTOLISSI, 2000), criar novos métodos heurísticos para a solução do problema, obtendo soluções melhores com qualidade e com tempo computacional aceitável.

O principal aspecto a ser destacado, neste artigo, refere-se ao desenvolvimento de métodos heurísticos de fase única (SN-1 e SN-2), que demonstraram, através da experimentação computacional, ser superiores que o melhor método heurístico composto de três fases.

8. REFERÊNCIAS BIBLIOGRÁFICAS

ALDOWAISAN, T.; ALLAHVERDI, A. New heuristics for m-machine no-wait flowshop to minimize total completion time. **OMEGA – The International Journal of Management Science**, v. 32, pp. 345-352, 2004.

BERTOLISSI, E. A simple no-wait flowshop scheduling heuristic for the no-wait flow-shop problem. 15th International Conference on Computer-Aided Production Engineering – CAPE´99, **Proceedings**, University of Durham Publishers, pp. 750-755, 1999.

BERTOLISSI, E. Heuristic algorithm for scheduling in th no-wait flow-shop. **Journal of Materials Processing Technology**, v. 107, pp. 459-465, 2000.

BONNEY, M. C.; GUNDRY, S.W. Solutions to the constrained flowshop sequencing problem. **Operations Research Quarterly**, v. 24, pp. 869-883, 1976.

CHEN, C.; NEPPALLI, R. V.; ALJABER, N. Genetic algorithms applied to the continuous flow shop problem. **Computers Industrial Engineers**, v. 30, pp. 919-929, 1996.

DEMAN, J. M. V.; BAKER, K. R. Minimizing mean flowtime in the flow shop with no intermediate queues. **AIIE Transactions**, v. 6, pp. 28-34, 1974.

FINK, A.; VOβ, S. Solving the continuous flow-shop scheduling problem by metaheuristics. **European Journal of Operational Research**, v. 151, pp. 400-414, 2003.

KING, R.; SPACHIS, A.S. Heuristics for flowshop scheduling. **International Journal of Production Research**, v. 18, pp. 343-357, 1980.

NAWAZ, M.; ENSCORE JR., E.E.; HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. **OMEGA – The International Journal of Management Science**, v.11, pp. 91-95, 1983.

RAJENDRAN, C.; CHAUDHURI, D. Heuristic algortihms for continuous flow-shop problem. **Naval Research Logistics**, v. 37, pp. 695-705, 1990.