REDUÇÃO DO ESTOQUE EM PROCESSAMENTO EM SISTEMAS DE PRODUÇÃO *FLOW SHOP* SEM INTERRUPÇÃO NA EXECUÇÃO DAS TAREFAS

REDUCING IN-PROCESS INVENTORY IN NO-WAIT FLOW SHOP PRODUCTION ENVIRONMENT

Lucas Yamada Scardoelli
Escola de Engenharia de São Carlos – USP
Departamento de Engenharia de Produção
13560-970 Caixa Postal 359 São Carlos-SP
E-mail: scarty@terra.com.br

Marcelo Seido Nagano
Escola de Engenharia de São Carlos – USP
Departamento de Engenharia de Produção
13560-970 Caixa Postal 359 São Carlos-SP
E-mail: drnagano@usp.br

RESUMO

Este artigo trata do problema de Programação de Operações em um ambiente de produção *Flow Shop*, tendo como objetivo minimizar o estoque em processamento, sem interrupção na execução das tarefas. Um novo método heurístico é proposto para a solução do problema. Por meio de uma experimentação computacional, o desempenho do método proposto é avaliado e comparado com os melhores métodos heurísticos reportados na literatura. Os resultados experimentais mostram a superioridade do novo método para o conjunto de problemas tratados.

PALAVRAS CHAVES: No-wait Flow shop. Estoque de processo. Métodos heurísticos. Área de classificação principal: Administração e Gestão da Produção.

ABSTRACT

This paper deals with the Permutation Flow Shop scheduling problem with the objective of minimizing total flow time such that jobs may not have their execution interrupted on or between machines after they have been started, therefore reducing in-process inventory. A new heuristic method is proposed for the scheduling problem solution. The proposed heuristic is compared with the best heuristics reported in the literature. Experimental results show that the new heuristic provides better solutions regarding both the solution quality and computational effort.

KEYWORDS: No-wait Flow shop. In-process inventory. Heuristics.

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.e., 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.

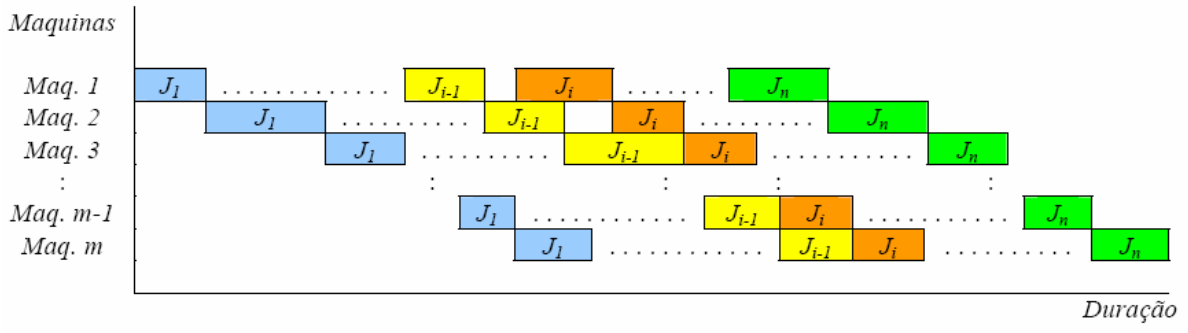

Figura 1 - *No-Wait Flow Shop* com m máquinas e n tarefas

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 ($1 \le g \le m-1$), 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 no início do processamento da tarefa que ocupa a primeira posição na seqüência na primeira máquina.

Devido a sua natureza, o problema *NWFS* é considerado como *NP-hard* para o caso de três ou mais máquinas (Chen et al., 1996).

Segundo Fink e Voβ (2003) em um problema *NWFS* geralmente verifica-se a ocorrência de um *delay* (d_{uv}) na primeira máquina, entre o início da tarefa u e o início da tarefa v, quando a tarefa v é processada diretamente após a tarefa u, onde o cálculo do *delay* pode ser efetuado pela seguinte expressão:

$$d_{uv} = t_{u1} + \max_{k}\left(\sum_{p=2}^{k} t_{up} - \sum_{p=1}^{k-1} t_{vp}, 0 \right) \qquad (1)$$

O objetivo deste trabalho é apresentar um novo método heurístico para o problema de programação de operações em ambientes *NWFS* com critério de minimização do tempo total de fluxo, verificando o seu desempenho comparado com os melhores métodos heurísticos existentes na literatura.

Para o problema apresentado, o tempo total de fluxo pode ser obtido conforme a seguinte expressão:

$$F = \sum_{i=2}^{n} (n+1-i) d_{[i-1][i]} + \sum_{i=1}^{n} \sum_{k=1}^{m} t_{ik} \qquad (2)$$

Resumidamente, esta expressão fornece a soma dos tempos de processamento das tarefas em todas as máquinas e a soma dos *delays* entre as tarefas seqüenciadas.

As características essenciais a serem consideradas para o novo método proposto serão: equilíbrio entre a qualidade da solução e eficiência computacional e; simplicidade e facilidade de implementação.

2. Problema de Programação de Operações *NWFS*

Deman e Baker (1974) foram alguns dos pioneiros a estudar o problema de ***NWFS***, com o critério de minimização do tempo total de fluxo. Em sua pesquisa eles apresentaram um algoritmo *branch and bound* para estabelecer todas as seqüências parciais, considerando a utilização de limitantes inferiores. Os autores concluíram que os resultados alcançados foram satisfatórios e que o problema podia ser solucionado tão rapidamente quanto os problemas tradicionais que utilizavam como critério a minimização do *makespan*.

Rajendran e Chaudhuri (1990) apresentaram dois algoritmos heurísticos construtivos, compostos de duas fases: uma primeira fase de ordenação inicial das tarefas e uma outra de construção da seqüência final avaliando seqüências parciais, a fim de estabelecer a seqüência final das tarefas. A experimentação computacional demonstrou que as soluções obtidas eram melhores, quando comparadas com os métodos anteriormente propostos por Bonney e Gundry (1976) e King e Spachis (1980) com o critério de minimização do *makespan.*

Chen *et al*. (1996) desenvolveram uma heurística, baseada no algoritmo genético, e, através da sua parametrização, conseguiram melhorar as soluções obtidas pelo método construtivo proposto por Rajendran e Chaudhuri (1990).

Bertolissi (1999) apresentou um método heurístico construtivo 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 em seguida é obtida uma ordenação das tarefas. Os experimentos computacionais mostraram que o método proposto não obtém soluções melhores que os métodos de Rajendran e Chaudhuri (1990).

Bertolissi (2000), apresenta um novo método heurístico construtivo composto de duas fases. Na primeira fase, similar a Bertolissi (1999), 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 mesmo procedimento de inserção de tarefas de Rajendran e Chaudhuri (1990). Os resultados dos experimentos computacionais permitem concluir que o método heurístico construtivo proposto obtém soluções melhores quando comparado aos métodos heurísticos construtivos existentes.

Fink e Voβ (2003) utilizaram os procedimentos metaheurísticos de Busca Tabu e *Simulated Annealing* e concluíram que a efetividade dos procedimentos metaheurísticos utilizados na solução do problema depende 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 adequado.

Aldowaisan e Allahverdi (2004) propuseram novos métodos heurísticos compostos de três fases. Segundo os resultados dos experimentos, 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, 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 de total de fluxo na ordenação parcial; na segunda, é obtida uma solução inicial semelhante ao método de inserção proposto por Nawaz *et al.* (1983); e na última, é aplicado o procedimento de melhoria utilizando a permutação de pares de tarefas.

3. Método heurístico proposto

O método heurístico proposto neste trabalho possui duas fases: na primeira é realizada uma ordenação inicial das tarefas utilizando-se da expressão 3; e na segunda busca-se construir a seqüência solução com a inserção de tarefas nas seqüências parciais e sua melhoria com um método de busca na vizinhança. Para fins de comparação o referido método será denominado **S&N**.

O método heurístico proposto por Bertolissi (2000) apresenta uma expressão que o próprio autor considera como a base do seu método proposto. Trata-se 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 termino de J_j) para m máquinas.

Dessa forma, segundo Bertolissi (2000), é possível calcular as parcelas das somas dos tempos de fluxo para o conjunto de pares de tarefas adjacentes para m máquinas com a expressão:

$$F_m(J_i, J_j) = 2t_{i1} + \sum_{k=2}^{m} t_{ik} + R_m(J_i, J_j) \quad (3)$$

Onde:

$$R_1(J_i, J_j) = t_{j1},$$

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

Na expectativa de minimizar o tempo total de fluxo, a expressão 3 possibilita a criação de uma ordenação inicial das tarefas a partir das menores parcelas das somas dos tempos de fluxo de pares de tarefas adjacentes.

A adoção desta forma de ordenação inicial é baseada no fato de que, para minimização do tempo total de fluxo de uma determinada seqüência de tarefas, deve-se ordenar as tarefas que apresentam menores tempos de processamentos. Porém, para atender ao caso particular de *NWFS*, é necessário considerar também o *delay* entre o início de determinada tarefa e o início da tarefa subseqüente.

1ª Fase - Ordenação inicial das tarefas

Assim, seja $J = \{J_1, J_2, ..., J_n\}$ um conjunto de n tarefas que devem ser processadas, e S o conjunto das tarefas programadas.

Passo 1

$J = \{J_1, J_2, ..., J_i, ..., J_n\}$;
$S = \emptyset$;
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 $J_v \in J$;
$S \leftarrow S \cup \{J_v\}$;
$J \leftarrow J - \{J_v\}$;
$u \leftarrow J_{[k]}$;
$k \leftarrow k + 1$;

Se $J \neq \emptyset$, volte para o ***Passo 2***. Caso contrário (ordenação inicial das tarefas concluída) vá para o ***Passo 3***.

2ª Fase - Construção da seqüência final

A construção da seqüência final é uma modificação do algoritmo proposto por Framinan e Leisten (2003) para o problema *NWFS*.

A modificação ocorre na segunda fase do algoritmo, onde a cada iteração é aplicado um procedimento de melhoria na seqüência parcial utilizando a vizinhança de inserção de tarefas.

Esta estrutura possibilita avaliar um maior número de seqüências parciais comparada ao método proposto por Framinan e Leisten (2003) não inviabilizando o tempo de computação.

A segunda parte do algoritmo proposto é definida pelos seguintes passos:

Passo 3

Selecione as duas primeiras tarefas da ordenação inicial S;

Passo 4

Para $L = 3$ a n;

- Selecione a tarefa que ocupa a $L - ésima$ posição na ordenação inicial S;
- Examine as L possibilidades de inserir a tarefa na seqüência parcial até então obtida, adotando aquela que leva a menor soma dos tempos de fluxo;
- Considerando toda a Vizinhança de Inserção da seqüência parcial com L tarefas, constituída de $(k-1)^2$ seqüências, determine a seqüência $S`$ associada a menor soma dos tempos de fluxo;
- Atualize, com a seqüência $S`$ as L primeiras posições da seqüência S.

Após a apresentação dos métodos heurísticos existentes e a proposição de um novo método heurístico construtivo de duas fases, será apresentada na próxima seção, a experimentação computacional.

4. Experimentação Computacional

O método heurístico proposto (S&N) foi comparado com os melhores métodos heurísticos reportados na literatura, ou seja, **Raj1** e **Raj2,** desenvolvidos por Rajendran e Chaudhuri (1990), **Bert,** de Bertolissi (2000), e 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 de problemas.

O terceiro grupo foi constituído por 2.000 problemas, subdivididos em 20 classes, considerados de grande porte, com número de tarefas $n \in \{90,100,110,120,130\}$ e $m \in \{5,10,15,20\}$.

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 $[1,99]$. 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. Para facilitar a visualização dos resultados, eles são apresentados em forma de gráficos.

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

O Desvio Relativo (DR_h) quantifica o desvio que o método h obtém em relação ao melhor tempo total de fluxo obtido para um mesmo problema, sendo calculado como segue:

$$DR_h = \left(\frac{F_h - F_*}{F_*} \right) \qquad (4)$$

Onde , F_h = Tempo total de fluxo obtido pelo método h;

F_* = Melhor tempo total de fluxo obtido pelos métodos, para um determinado problema.

O tempo médio de computação (em segundos) é calculado pela soma dos tempos de processamento em cada problema, dividido pelo número total de problemas resolvidos.

Todos os resultados apresentados foram agrupados com relação ao número de máquinas possibilitando uma análise global dos resultados.

6. Análise dos Resultados

Os resultados obtidos foram analisados em três partes na seguinte ordem: Porcentagem de Sucesso; Porcentagem de Desvio Médio Relativo; e Tempo Médio de Computação.

- **Análise da Porcentagem de Sucesso**

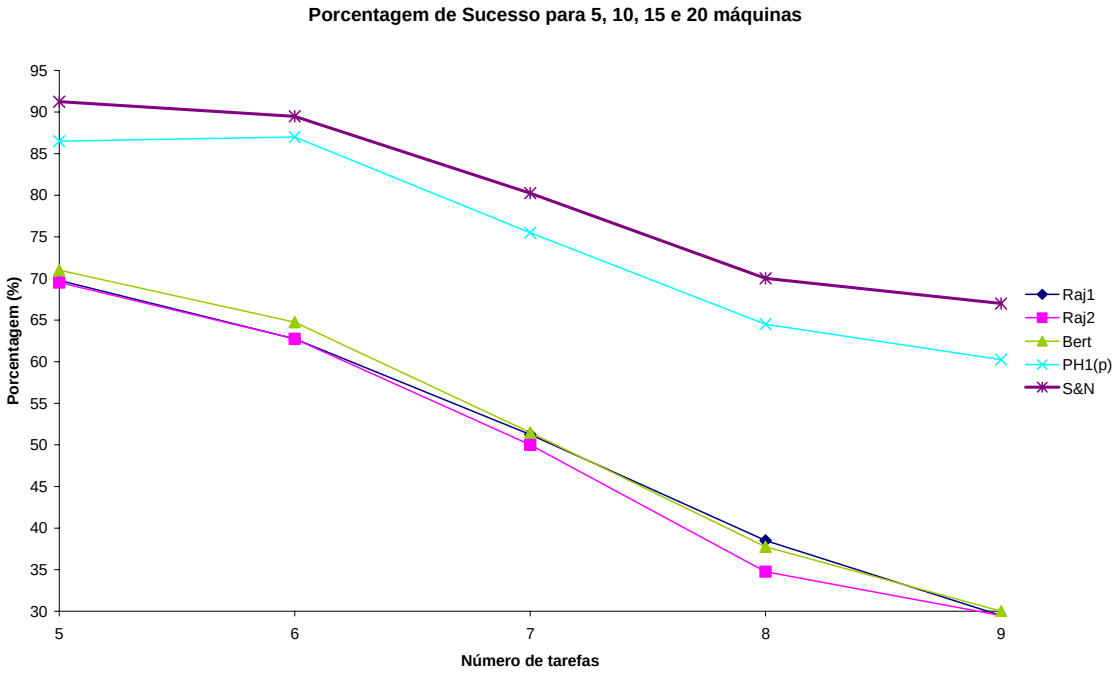

Figura 2 - Porcentagem de sucesso para problemas de 5, 10, 15 e 20 máquinas agrupadas (pequeno porte)

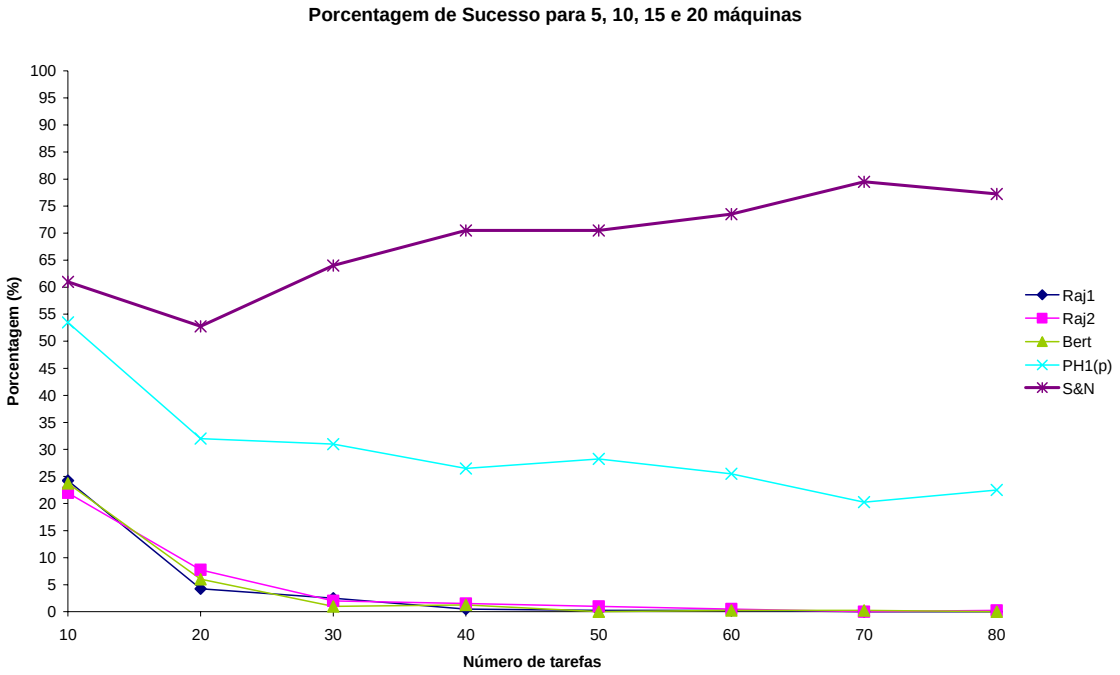

Figura 3 - Porcentagem de sucesso para problemas de 5, 10, 15 e 20 máquinas agrupadas (médio porte)

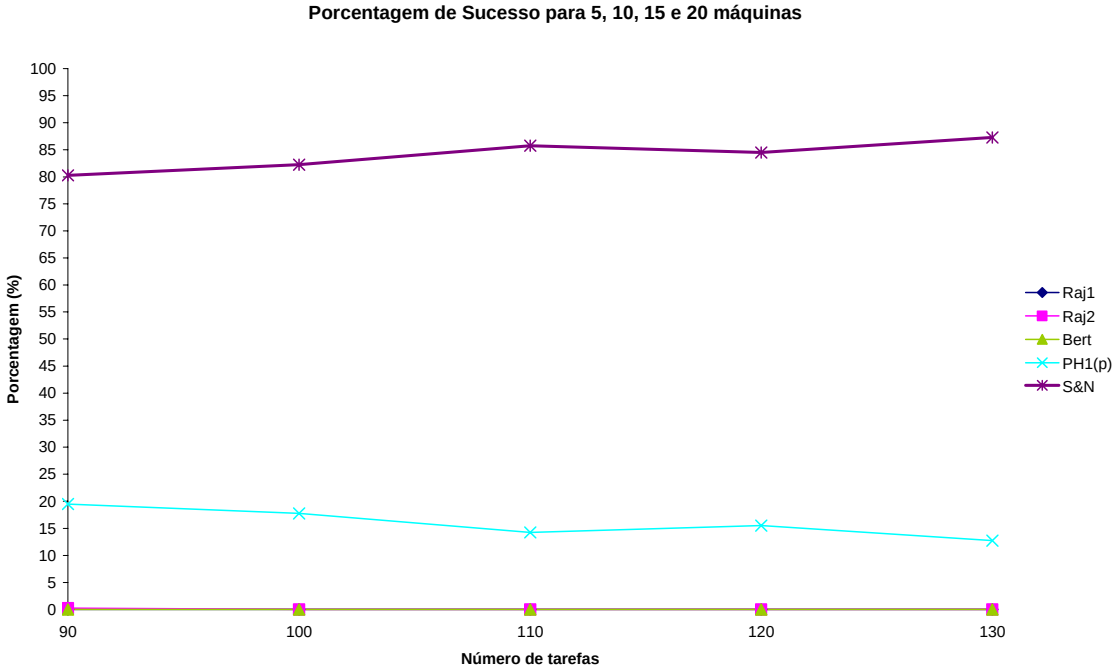

Figura 4 - Porcentagem de sucesso para problemas de 5, 10, 15 e 20 máquinas agrupadas (grande porte)

Em uma primeira análise verifica-se que a porcentagem de sucesso do método S&N é superior a todos os outros métodos, em todos os casos de variação de 5 a 130 tarefas.

Além disso, é possível concluir que a partir de problemas com 20 tarefas, o método S&N apresenta tendência de aumento da porcentagem de sucesso à medida que aumenta o número de tarefas.

Por fim, fica consolidado que independente do número de máquinas, o método S&N é

superior aos demais existentes, principalmente para problemas de grande porte.

Ainda com relação aos problemas de grande porte, verifica-se que o segundo melhor método é o PH1(p) que apresenta porcentagem de sucesso inferior a 20% em todos os casos, enquanto o melhor método apresenta porcentagem de sucesso superior de 80%.

- **Análise da Porcentagem de Desvio Médio Relativo**

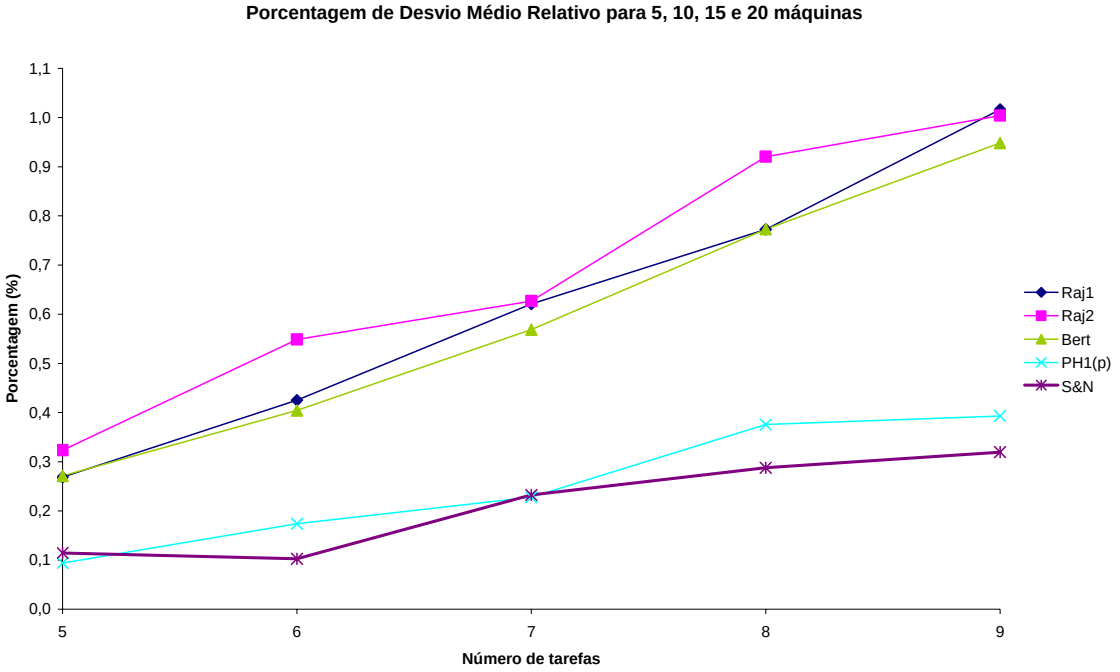

Figura 5 - Porcentagem de desvio médio relativo para problemas de 5, 10, 15 e 20 máquinas agrupadas (pequeno porte)

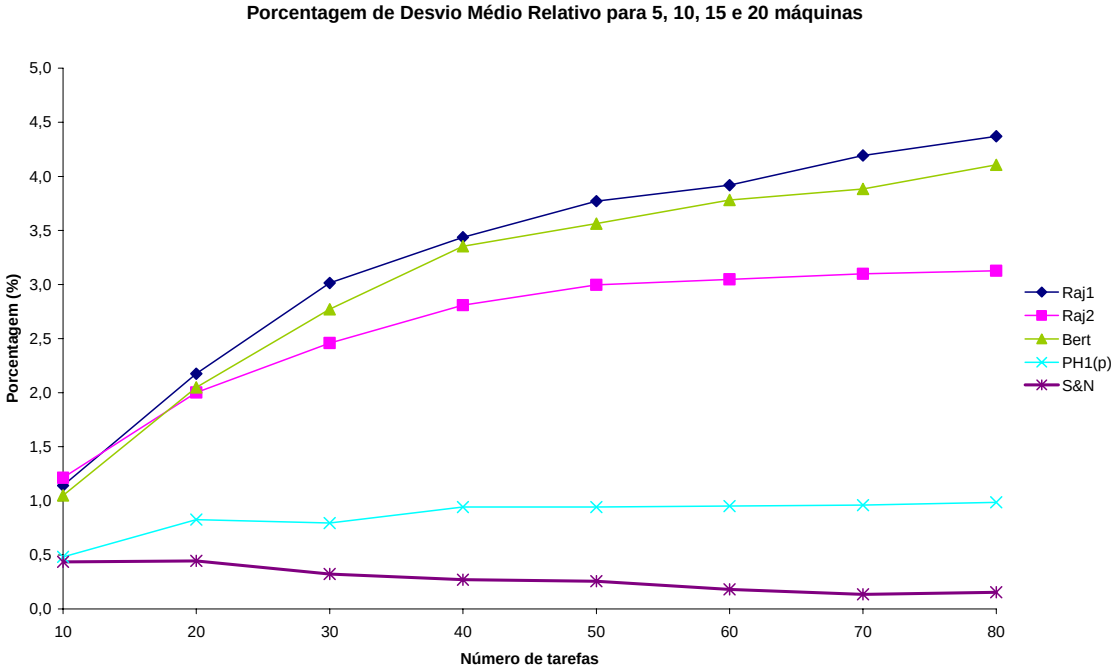

Figura 6 - Porcentagem de desvio médio relativo para problemas de 5, 10, 15 e 20 máquinas agrupadas (médio porte)

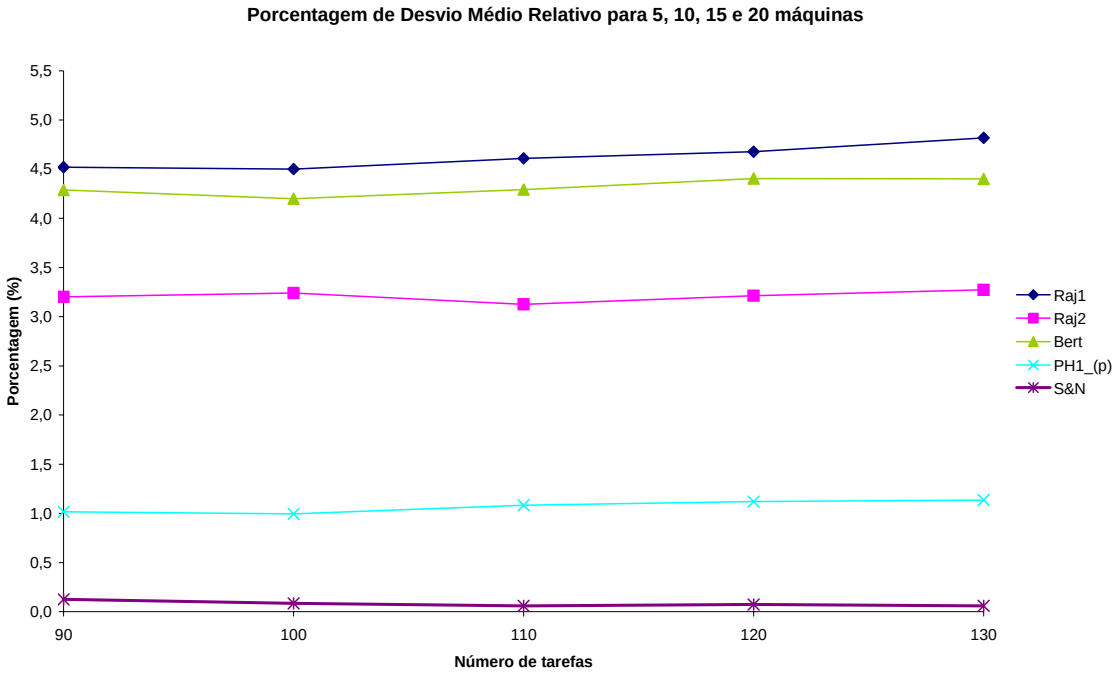

Figura 7 - Porcentagem de desvio médio relativo para problemas de 5, 10, 15 e 20 máquinas agrupadas (grande porte)

O desvio médio relativo para os problemas de pequeno porte apresenta-se muito próximo para os métodos Raj1, Raj2 e Bert, porém os métodos S&N e PH1(p) apresentam menor desvio médio relativo que os demais métodos, principalmente devido ao fato de apresentarem soluções de melhor qualidade.

A partir dos problemas com 20 tarefas, o algoritmo S&N destaca-se, demonstrando um desvio médio relativo inferior aos demais algoritmos, principalmente em relação aos algoritmos Raj1, Raj2 e Bert.

Verifica-se na figura 7 que, para os problemas de grande porte, a heurística S&N também é a que apresenta menor porcentagem de desvio médio relativo (próximo de zero), enquanto o segundo melhor método apresenta uma porcentagem de desvio médio relativo acima de 1%. Em relação aos métodos compostos de duas fases, o método S&N apresentou soluções pelo menos 3% melhores, como pode ser observado também na figura 7.

- **Tempo Médio de Computação**

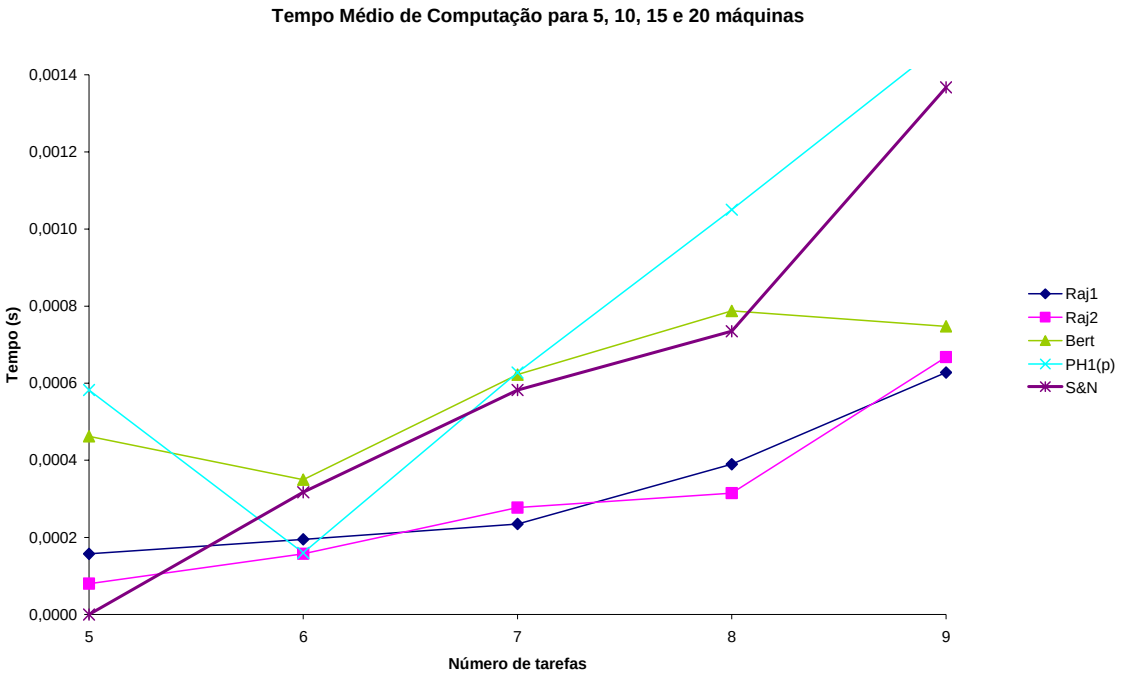

Figura 8 - Tempo Médio de Computação para problemas de 5, 10, 15 e 20 máquinas agrupadas (pequeno porte)

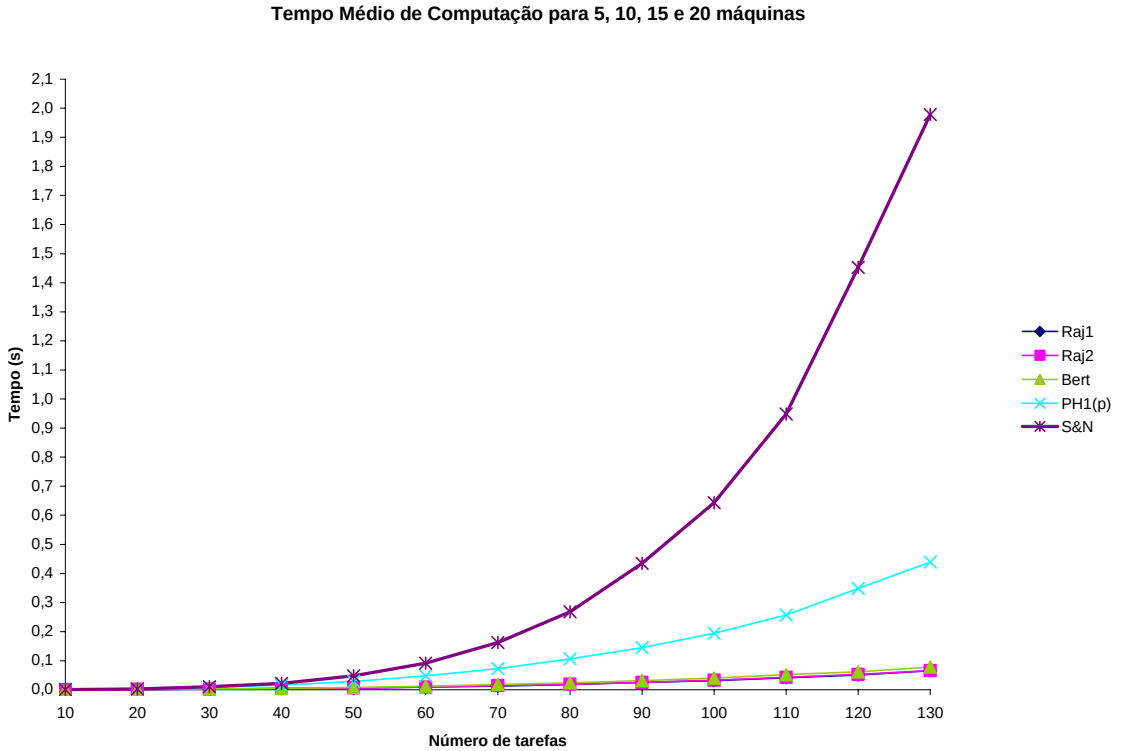

Figura 9 - Tempo Médio de Computação para problemas de 5, 10, 15 e 20 máquinas agrupadas (médio e grande porte)

O método S&N para os problemas de grande porte, assim como nos problemas de médio porte, tem uma característica particular em relação aos demais: quanto maior o número de tarefas,

maior o tempo de computação médio. Para problemas com 130 tarefas o tempo de processamento aproxima-se de 2 segundos, como pode ser observado na figura 9.

De forma geral, entre todos os métodos avaliados nesta experimentação, verifica-se que o método proposto S&N é o que apresenta melhores resultados tanto em termos de porcentagem de sucesso, quanto em termos de porcentagem de desvio médio relativo.

Percebe-se também que o tempo de computação médio do método proposto S&N possui um crescimento exponencial em relação ao número de tarefas. Porém, mesmo para casos extremos, o método exige um tempo computacional que não inviabiliza sua aplicação.

7. Considerações Finais

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

Porém, tendo em vista 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 procurou evidenciar que apesar dos algoritmos existentes proporcionarem boas soluções, é possível, por meio da propriedade já existente e apresentada em Bertolissi (2000), criar novos métodos de soluções para o problema.

O principal aspecto a ser destacado neste artigo refere-se ao fato de que os resultados experimentais mostraram que o método heurístico proposto S&N possui um desempenho superior aos demais, para solução do problema em questão.

A experimentação apresentou um tempo computacional maior em relação aos métodos já existentes, porém tal aumento é compensado pela facilidade de implementação do método e pela melhoria na qualidade das soluções.

Ainda com relação à experimentação computacional realizada, uma consideração interessante não pode esquecida. Quando avaliados separadamente os métodos Bert, Raj1 e Raj2, foi verificado que o método proposto por Bertolissi (2000) apenas equipara-se aos métodos propostos por Rajendran e Chaudhuri (1990) para problemas de pequeno porte, sendo inferior nos problemas de médio e grande porte. Assim, diferentemente da conclusão apresentada por Bertolissi (2000), seu método não é superior aos métodos Raj1 e Raj2.

8. Referências Bibliográficas

Aldowaisan, T. e Allahverdi, A. (2004). New heuristics for m-machine no-wait flowshop to minimize total completion time. *OMEGA – The International Journal of Management Science,* 32, 345- 352.

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

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

Bonney, M.C. e Gundry, S.W. (1976). Solutions to the constrained flowshop sequencing problem. *Operations Research Quarterly*, 24, 869-883.

Chen, C., Neppalli, R.V. e Aljaber, N. (1996). Genetic algorithms applied to the continuous flow shop problem. *Computers Industrial Engineers*, 30, 4, 919-929.

Deman, J.M.V. e Baker, K.R. (1974). Minimizing mean flowtime in the flow shop with no intermediate queues. *AIIE Transactions*, 6, 1, 28-34.

Fink, A. e Voβ, S. (2003). Solving the continuous flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research*, 151, 400-414.

Framinan, J.M. e Leisten, R. (2003). An efficient constructive heuristic for flowtime minimisation in permutation flow shops. *OMEGA – The International Journal of Management Science*, 31, 311-317.

King, R. e Spachis, A.S. (1980). Heuristics for flowShop scheduling. *International Journal of Production Research*, 18, 343-357.

Nawaz, M., Enscore Jr., E.E. e Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA – The International Journal of Management Science*, 11, 1, 91-95.

Rajendran, C. e Chaudhuri D. (1990). Heuristic algortihms for continuous flow-shop problem. *Naval Research Logistics*, 37, 695-705.