PROGRAMAÇÃO DA PRODUÇÃO EM SISTEMAS *FLOW SHOP* UTILIZANDO UM MÉTODO HEURÍSTICO HÍBRIDO ALGORITMO GENÉTICO-*SIMULATED ANNEALING*

Walther Rogério Buzzo
João Vitor Moccellin
Escola de Engenharia de São Carlos – USP
E-mail: jvmoccel@sc.usp.br

Resumo

Este artigo trata do problema de programação de tarefas flow shop *permutacional. Diversos métodos heurísticos têm sido propostos para tal problema, sendo que um dos tipos de método consiste em melhorar soluções iniciais a partir de procedimentos de busca no espaço de soluções, tais como Algoritmo Genético (AG) e* Simulated Annealing *(SA). Uma idéia interessante que tem despertado gradativa atenção refere-se ao desenvolvimento de métodos heurísticos híbridos utilizando Algoritmo Genético e Simulated Annealing. Assim, o objetivo é combinar as técnicas de tal forma que o procedimento resultante seja mais eficaz do que qualquer um dos seus componentes isoladamente. Neste artigo é apresentado um método heurístico híbrido Algoritmo Genético-*Simulated Annealing *para minimizar a duração total da programação* flow shop *permutacional. Com o propósito de avaliar a eficácia da hibridização, o método híbrido é comparado com métodos puros AG e SA. Os resultados obtidos a partir de uma experimentação computacional são apresentados.*

Palavras-chave: programação da produção,* flow shop *permutacional, metaheurísticas híbridas.

1. Introdução

O problema de programação de operações em um ambiente *Flow Shop* é um problema de programação de produção no qual **n** tarefas devem ser processadas por um conjunto de **m** máquinas distintas, tendo o mesmo fluxo de processamento nas máquinas. Usualmente a

solução do problema consiste em determinar uma seqüência das tarefas dentre as (n!) seqüências possíveis, que é mantida para todas as máquinas (Programação Permutacional) e que procura otimizar uma determinada Medida de Desempenho da programação, geralmente associada ao fator tempo. Neste trabalho adota-se como função-objetivo a Duração Total da Programação (*makespan*), ou seja, o intervalo de tempo entre o início de execução da primeira tarefa na primeira máquina e o término de execução da última tarefa na última máquina.

As hipóteses usuais do problema de Programação de Tarefas *Flow Shop* são:

1) cada máquina está disponível continuamente, sem interrupções;
2) cada máquina pode processar apenas uma tarefa de cada vez;
3) cada tarefa pode ser processada por uma máquina de cada vez;
4) os tempos de processamento das tarefas nas diversas máquinas são determinados e fixos;
5) as tarefas têm a mesma data de liberação, a partir da qual, qualquer uma pode ser programada e executada;
6) os tempos de preparação das operações nas diversas máquinas são incluídos nos tempos de processamento e independem da seqüência de operações em cada máquina e
7) as operações nas diversas máquinas, uma vez iniciadas não devem ser interrompidas.

Na teoria que estuda a complexidade dos problemas de natureza combinatorial, o problema em questão é classificado como *NP-hard* (GAREY *et al*., 1976), de forma que pode ser resolvido eficientemente de maneira ótima somente em casos de pequeno porte. Resolver um problema de otimização combinatorial consiste em se encontrar uma boa solução ou a solução ótima dentre um número finito de soluções possíveis. Por exemplo, um problema de programação *Flow Shop* Permutacional envolvendo apenas 10 tarefas apresenta 3.628.800 soluções possíveis.

Nas últimas quatro décadas, um extenso esforço de pesquisa tem sido dedicado ao problema. Técnicas de Programação Matemática, tais como Programação Linear Inteira (SELEN & HOTT, 1986) e técnicas de enumeração do tipo *branch-and-bound* (IGNALL & SCHRAGE, 1965), têm sido empregadas para a solução ótima do problema. Entretanto, tais técnicas não são eficientes em termos computacionais, em problemas de médio e grande porte. Assim, muitos métodos heurísticos têm sido propostos, os quais, de maneira geral, podem ser classificados em dois grupos: métodos construtivos e métodos de melhoria.

Os métodos heurísticos construtivos geram uma única seqüência das tarefas, a qual é adotada como solução final do problema, por exemplo: PALMER (1965), NEH (NAWAZ, ENSCORE & HAM, 1983).

Nos métodos heurísticos de melhoria, obtém-se uma solução inicial e posteriormente por meio de algum procedimento iterativo (geralmente envolvendo trocas de posições das tarefas na seqüência) busca-se obter uma seqüência das tarefas melhor que a atual quanto à medida de desempenho adotada. Nesta categoria, destacam-se os procedimentos de busca em vizinhança, como por exemplo DANNENBRING (1977), considerado um método de busca simples. Mais recentemente, foram desenvolvidos métodos de busca em vizinhança de maior complexidade (Busca Tabu e *Simulated Annealing*) que têm sido alvo de grande interesse na comunidade científica em função de aplicações bem sucedidas reportadas na literatura. Exemplos de aplicações dessas técnicas para o problema *Flow Shop* Permutacional são encontrados em: OSMAN & POTTS (1989); WIDMER & HERTZ (1989); OGBU & SMITH (1990); TAILLARD (1990); MOCCELLIN (1995); ISHIBUCHI, MISAKI & TANAKA (1995); ZEGORDI, ITOH & ENKAWA (1995); NOWICKI & SMUTNICKI (1996); PARK & KIM (1998) e MOCCELLIN & NAGANO (1998).

Outra técnica que pode ser considerada do tipo de melhoria, denominada Algoritmo Genético, tem despertado interesse pela sua capacidade de solução de problemas de natureza

combinatorial. REEVES (1995) utilizou o Algoritmo Genético para a minimização do *makespan* em um *Flow Shop* Permutacional.

Os métodos heurísticos Busca Tabu, *Simulated Annealing* e Algoritmo Genético (também conhecidos por Metaheurísticas) são procedimentos de busca no espaço de soluções, definidos por estratégias que exploram apropriadamente a topologia de tal espaço.O sucesso das metaheurísticas se deve a fatores como:

i) alusão a mecanismos de otimização da natureza (nos casos do Algoritmo Genético e do *Simulated Annealing*);
ii) aplicabilidade geral da abordagem;
iii) facilidade de implementação e
iv) qualidade da solução obtida aliada a um esforço computacional relativamente baixo.

Uma idéia interessante que tem recebido gradativa atenção refere-se ao desenvolvimento de métodos metaheurísticos híbridos utilizando Busca Tabu (BT), *Simulated Annealing* (*SA*) e Algoritmo Genético (AG). Assim, o objetivo é combinar as técnicas BT, *SA* e AG, preservando suas características de ação "inteligente", de tal forma que o procedimento resultante seja mais eficaz do que qualquer um dos seus componentes isoladamente.

Na literatura já foram reportados alguns trabalhos relatando experimentações com variantes ou combinações desses três métodos, desenvolvidos para resolverem diversos problemas. Por exemplo: KURODA & KAWADA (1994); KIM, NARA & GEN (1994); GLOVER, KELLY & LAGUNA (1995); ROACH & NAGI (1996); PIRLOT (1996).

Especificamente relacionados com o problema de Programação de Tarefas *Flow Shop*, são encontrados na literatura: MURATA, ISHIBUCHI & TANAKA (1996), os quais examinaram duas hibridizações do Algoritmo Genético com outros algoritmos de busca; DÍAZ (1996), que utilizou o *Simulated Annealing* para obter uma solução a ser melhorada pela Busca Tabu; MOCCELLIN & SANTOS (2000) e SOUZA & MOCCELLIN (2000), que desenvolveram métodos híbridos combinando Busca Tabu-*Simulated Annealing* e Algoritmo Genético-Busca Tabu, respectivamente.

Neste artigo é apresentado um método híbrido para o Problema de Programação de Tarefas *Flow Shop* Permutacional buscando a minimização do *makespan*, utilizando-se as metaheurísticas Algoritmo Genético e *Simulated Annealing*, completando assim, com relação aos dois últimos trabalhos mencionados acima, as três combinações de pares das metaheurísticas BT, *SA* e AG.

O método híbrido proposto, bem como os métodos puros *Simulated Annealing* e Algoritmo Genético que foram utilizados na sua concepção, são apresentados na próxima seção.

2. O Método Metaheurístico Híbrido Proposto

O método híbrido (denominado HBGASA) apresentado neste artigo, tem como ponto de partida duas soluções iniciais geradas por dois métodos heurísticos construtivos que fornecem soluções de boa qualidade. A seguir, aplica-se nas soluções iniciais ("pais-sementes") o operador de cruzamento do Algoritmo Genético puro, selecionando-se dentre as quatro soluções disponíveis ("pais e filhos") as duas que apresentam os melhores valores da medida de desempenho. Na seqüência do método HBGASA, aplica-se o procedimento *Simulated Annealing* às duas melhores soluções, separadamente. Como resultado do procedimento *SA*, são obtidas duas novas soluções, sobre as quais é aplicado novamente o operador de cruzamento do Algoritmo Genético puro. Repete-se o processo de busca de novas e eventualmente melhores soluções, até ser atingida a Condição de Parada do método.

A concepção do método híbrido HBGASA teve como base os princípios de uma heurística, ou seja, simplicidade, facilidade de implementação aliada a um pequeno esforço computacional e eficácia quanto à qualidade da solução a partir do conhecimento da relação entre a estrutura do problema e a medida de desempenho adotada. Para tanto, foram utilizados os resultados

positivos de pesquisas anteriores relativas ao Algoritmo Genético (MOTA, 1996) e à técnica *Simulated Annealing* (MOCCELLIN, 1994).

Os resultados quanto ao Algoritmo Genético conduziram à opção por um algoritmo simples, com uma pequena população para seleção dos "pais" em cada iteração, porém com boas características "genéticas", uma vez que a força de um Algoritmo Genético baseia-se na hipótese de que a melhor solução se encontra em regiões do espaço de soluções viáveis que contêm uma grande proporção relativa de boas soluções e que tais regiões podem ser identificadas por uma amostragem minuciosa do espaço de soluções.

O procedimento *Simulated Annealing* (denominado ModSAfshopH) é uma extensão melhorada do método SAfshopH (MOCCELLIN, 1994), o qual foi comparado com o tradicional método SA(S,R), proposto por OSMAN & POTTS (1989), obtendo a melhor solução em 69,2% dos problemas testados.

A descrição detalhada do método híbrido é apresentada a seguir.

2.1 Parâmetros Quantitativos e Não Quantitativos do Método Híbrido HBGASA

Soluções Iniciais

As soluções iniciais são fornecidas pelo método heurístico construtivo NEH (NAWAZ, ENSCORE & HAM, 1983) e pela solução inicial do método FSHOPH (MOCCELLIN, 1995). No método *Simulated Annealing* puro que é comparado com o híbrido HBGASA, a solução inicial é a melhor dentre as soluções iniciais do HBGASA.

Vizinhança

Na pesquisa relatada neste trabalho**,** foi utilizado o tipo de Vizinhança de Inserção (*Shift Neighborhood*)*,* no qual uma seqüência vizinha é obtida da seqüência atual removendo-se uma tarefa de sua posição e inserindo-a em uma outra posição. Para um conjunto de n tarefas, o tamanho da Vizinhança de Inserção é dado por $(n-1)^2$.

Formas de Busca na Vizinhança

Foi empregada a busca parcial aleatória, que consiste no exame parcial da vizinhança, ou seja, apenas uma parcela Nvp(S) do número total de vizinhos é avaliada, a qual é gerada aleatoriamente.

A parcela da vizinhança, baseada no trabalho desenvolvido por NAGANO (1995), é dada pelas seguintes expressões:

$$Nvp(S) = \mathbf{p}\,(n-1)^2, \text{ para } n \leq 30$$

$$Nvp(S) = \mathbf{p}\,[1741 - 900 \exp(-0{,}04(n-30))], \text{ para } n > 30$$

onde **p** é um parâmetro a ser obtido experimentalmente no conjunto {0,05; 0,10; 0,15; 0,20; 0,25; ..... 0,90; 0,95; 1,00} e **n** é o número de tarefas a serem programadas.

Condição de Parada

O método híbrido encerra a busca por melhores soluções após um número TNbS(m,n) predeterminado de seqüências avaliadas de acordo com a Tabela 1. Tais números foram obtidos por experimentação prévia: para cada classe de problema (número de máquinas **m**, número de tarefas **n**), o valor TNbS(m,n) corresponde ao número médio de seqüências avaliadas utilizando-se o método heurístico de Busca Tabu FShop.TS5 (MOCCELLIN & NAGANO, 1998).

Convém ressaltar que os dois métodos puros (algoritmo genético e *Simulated Annealing*), que são comparados com o híbrido, utilizam a mesma condição de parada.

Operador de Reprodução

A reprodução é efetuada sobre a população constituída pelas duas seqüências "pais" e duas seqüências "filhos". As duas seqüências que apresentarem as menores durações totais de programação são selecionadas para a aplicação do operador de cruzamento.

Operador de Cruzamento

Foram selecionados dois tipos de operadores de cruzamento, a partir dos resultados reportados

Tabela 1 – Condição de Parada.

Número de (máquinas, tarefas)	TNbS(m,n)	Número de (máquinas, tarefas)	TNbS(m,n)	Número de (máquinas, tarefas)	TNbS(m,n)
(4, 20)	9693	(7, 20)	10812	(10, 20)	16750
(4, 30)	43732	(7, 30)	59666	(10, 30)	73503
(4, 40)	77742	(7, 40)	162358	(10, 40)	125728
(4, 50)	105623	(7, 50)	165856	(10, 50)	154223
(4, 60)	133502	(7, 60)	178017	(10, 60)	166257
(4, 70)	137452	(7, 70)	135330	(10, 70)	173862
(4, 80)	166050	(7, 80)	107244	(10, 80)	189378
(4, 90)	155455	(7, 90)	128733	(10, 90)	189406
(4, 100)	194680	(7, 100)	108136	(10, 100)	190884

por MOTA (1996). Tais operadores mostraram-se os mais eficazes dentre os operadores de cruzamento analisados no referido trabalho. Tais operadores são conhecidos pelas denominações: *One Point Crossover* ou Operador de Cruzamento de Um Corte (1X), e *Partially Matched Crossover* ou Operador de Cruzamento PMX.

Operador de Mutação

Este operador é necessário para a introdução e a manutenção da diversidade genética da população. Apesar da sua relativa importância, ele não é utilizado no método híbrido HBGASA, uma vez que nesse método tal função é realizada pelo procedimento *Simulated Annealing*.

Porém, para o Algoritmo Genético puro (a ser comparado com o método híbrido) foram desenvolvidos 3 operadores de mutação, a saber:

i) operador mutação 1: consiste em escolher aleatoriamente uma tarefa e trocá-la de posição com a sua subseqüente;
ii) operador mutação 2: troca aleatoriamente as posições de duas tarefas quaisquer da seqüência e
iii) operador mutação 3: escolhe ao acaso um ponto para se dividir a seqüência em duas subseqüências, as quais têm suas posições relativas invertidas.

Pode-se notar que os operadores de mutação propostos têm graus de "perturbação" crescentes. O operador mutação 1 é aplicado toda vez que não houver melhoria na melhor seqüência até então obtida, após 0,01TNbS(m,n) seqüências sucessivas geradas e avaliadas. O mutação 2 é utilizado se não houver melhoria após 0,027TNbS(m,n) seqüências sucessivas. O operador mutação 3, que causa uma maior modificação na seqüência gerada é aplicado após 0,10TNbS(m,n) seqüências sucessivas sem melhoria quanto à melhor solução até o momento obtida. Os coeficientes 0,01; 0,027 e 0,10 foram estabelecidos em função do "grau de perturbação" de cada operador e também de forma que não haja a possibilidade de aplicação simultânea de dois ou dos três operadores.

Temperatura Inicial T_1

A temperatura inicial é um dos parâmetros sensíveis e experimentais do método híbrido HBGASA. Com o objetivo de obter o melhor valor para tal parâmetro, foi estabelecido um conjunto representativo dado por {20, 30, 45, 60, 70}.

Função de Resfriamento $F(T_k)$ (Cooling Schedule)

No procedimento *Simulated Annealing*, após cada iteração ocorre uma diminuição da

temperatura conforme a seguinte expressão, sugerida por OSMAN & POTTS (1989):

$$T_{k+1} = T_k / (1 + \beta T_k)$$

onde:

- T_k = temperatura da k-ésima iteração;
- β é um parâmetro dado pela relação: $\beta = (T_1 - T_K) / [(K - 1).(T_1 . T_K)]$, sendo:
  - T_1 = temperatura inicial;
  - T_K = temperatura final e
  - K = número de iterações.

O número de iterações K depende da estrutura do método, puro ou híbrido, sendo uma função da Condição de Parada. A temperatura final, assim como no trabalho de OSMAN & POTTS (1989), é adotada igual a 1 (um).

Probabilidade de Aceitação p(k)

A probabilidade de aceitação de um "movimento" no procedimento *Simulated Annealing* é calculada pela distribuição de Boltzmann, ou seja:

$$p(k) = \exp (-\Delta / T_k)$$

onde:

- $\Delta = C(\sigma') - C(\sigma)$;
- $C(\sigma')$: *makespan* da solução "candidata", extraída da vizinhança da solução atual (corrente);
- $C(\sigma)$: *makespan* da solução atual e
- T_k: temperatura na iteração k.

É importante salientar, também, que o método híbrido proposto se diferencia dos métodos híbridos Algoritmo Genético – *Simulated Annealing* encontrados na literatura.

No trabalho de ROACH & NAGI (1996) foi proposto um algoritmo híbrido para o problema de montagem em múltiplos níveis (ambiente *Just In Time*), que executa o Algoritmo Genético e o *Simulated Annealing* em duas fases que se alternam: na sua primeira, o Algoritmo Genético gera as soluções iniciais aleatoriamente ou utilizando heurísticas conhecidas. Tais soluções são submetidas ao cruzamento e à seleção (para produzir novas soluções) e, somente após dez gerações, as soluções passam a ser melhoradas pelo *Simulated Annealing*, na sua segunda fase.

O método híbrido apresentado no presente trabalho também gera as soluções iniciais usando heurísticas conhecidas, além de utilizar o operador de cruzamento para gerar novas soluções. Porém, as duas melhores soluções são imediatamente submetidas ao *Simulated Annealing*.

No caso específico de problemas de programação de tarefas *Flow Shop,* tem-se o trabalho de MURATA, ISHIBUCHI & TANAKA (1996), que também propuseram um método heurístico híbrido Algoritmo Genético-*Simulated Annealing*. Eles utilizaram o Algoritmo Genético como base para o *Genetic Simulated Annealing* (*GSA*). Porém, a diferença entre o *GSA* e o método HBGASA é que no *GSA*, o *Simulated Annealing* é inserido entre os passos 'inicialização' (geração da população inicial) e 'seleção' realizados pelo Algoritmo Genético, enquanto que no método híbrido apresentado neste artigo, o Algoritmo Genético e o *Simulated Annealing* operam em fases distintas.

2.2 O Algoritmo Genético Puro (1xGA ou pmxGA) (operadores de cruzamento 1x ou pmx)

PASSO 1)

Seleção dos pais: obter a seqüência inicial 1 (solução do NEH) e a seqüência inicial 2 (solução inicial do FSHOPH).

s_1 := seqüência inicial 1
s_2 := seqüência inicial 2
$f(s_1)$:= *makespan* da seqüência inicial 1
$f(s_2)$:= *makespan* da seqüência inicial 2

Se $f(s_1) \leq f(s_2)$, então
  BS := s_1 {melhor solução já obtida}
e BM := $f(s_1)$ {*makespan* da melhor solução}
Caso contrário, BS := s_2 e BM := $f(s_2)$.

Enquanto o número total de seqüências geradas e avaliadas $\leq TNbS(m,n)$ *e* $s_1 \neq s_2$*:*

PASSO 2)

Aplicar o operador de cruzamento em s_1 e s_2:

$s_1 \times s_2 \rightarrow s_3$ e s_4 (onde s_3 e s_4 são as seqüências descendentes).

Aplicar o *operador de mutação* em s_3 e s_4, se for o caso.

Ordenar as 4 seqüências em ordem não decrescente do *makespan* f(s).

PASSO 3)

Selecionar as 2 seqüências que apresentam os 2 menores *makespans*

s_1 := seq[1] {seqüência com o primeiro melhor *makespan*}

$f(s_1)$:= f(seq[1]) {*makespan* da primeira melhor seqüência}

s_2 := seq[2] {seqüência com o segundo melhor *makespan*}

$f(s_2)$:= f(seq[2]) {*makespan* da segunda melhor seqüência}

Se $f(s_1) < BM$ então
BS := seq[1]
BM := f(seq[1]).

PASSO 4)

Tomar as seqüências s_1 e s_2 e aplicar o procedimento descrito a partir do *Passo 2*.

- *Resultados:*
  BS: a melhor seqüência das tarefas (solução do problema);
  BM: valor do *makespan* ou duração total da programação das tarefas segundo BS.

2.3 O Algoritmo *Simulated Annealing* Puro (ModSAfshopH)

PASSO 1)

Obtenção da solução inicial s: obter a seqüência 1 (solução do NEH) e a seqüência 2 (solução inicial do FSHOPH).

s_1 := seqüência 1
s_2 := seqüência 2
$f(s_1)$:= *makespan* da seqüência 1
$f(s_2)$:= *makespan* da seqüência 2

Se $f(s_1) \leq f(s_2)$, então $s := s_1$
Caso contrário, $s := s_2$
BS := s {melhor solução já obtida}
BM := f(s) {*makespan* da melhor solução}.

Enquanto o número total de seqüências geradas e avaliadas ≤ TNbS(m,n):

PASSO 2)

Obter aleatoriamente a seqüência s' a partir da vizinhança de inserção da seqüência atual s.

PASSO 3)

Se $f(s') \leq f(s)$ então s := s'

Caso contrário, tomar ao acaso um valor R do intervalo [0,1]

Se $R \leq p(k)$, então s := s'

Atualizar o valor do parâmetro temperatura.

PASSO 4)

Se $f(s) < BM$, então BS := s e BM := f(s).

Voltar ao *Passo 2*.

- *Resultados:*
  BS: a melhor seqüência das tarefas (solução do problema);
  BM: valor do *makespan* ou duração total da programação das tarefas segundo BS.

2.4 O Algoritmo Híbrido HBGASA

Com o objetivo de salientar a hibridização do HBGASA, os passos 1, 2 e 3 referem-se à sua "parte Algoritmo genético", enquanto que os *passos 4* e *5* (em *itálico*) correspondem à "parte *Simulated Annealing*".

PASSO 1)

Seleção dos pais: obter a seqüência inicial 1 (solução do NEH) e a seqüência inicial 2 (solução inicial do FSHOPH).

s_1 := seqüência inicial 1

s_2 := seqüência inicial 2

$f(s_1)$:= *makespan* da seqüência inicial 1

$f(s_2)$:= *makespan* da seqüência inicial 2

Se $f(s_1) \leq f(s_2)$, então

BS := s_1 {melhor solução já obtida}

e BM := $f(s_1)$ {*makespan* da melhor solução}

Caso contrário, BS := s_2 e BM := $f(s_2)$.

Enquanto o número total de seqüências geradas e avaliadas $\leq TNbS(m,n)$ *e* $s_1 \neq s_2$*:*

PASSO 2)

Aplicar o operador de cruzamento em s_1 e s_2:

$s_1 \times s_2 \rightarrow s_3$ e s_4 (onde s_3 e s_4 são as seqüências descendentes)

Ordenar as 4 seqüências em ordem não decrescente do *makespan* f(s).

PASSO 3)

Selecionar as 2 seqüências que apresentam os 2 menores *makespans*

s_1 := seq[1] {seqüência com o primeiro melhor *makespan*}

$f(s_1)$:= f(seq[1]) {*makespan* da primeira melhor seqüência}

s2 := seq[2] {seqüência com o segundo melhor *makespan*}

f(s2) := f(seq[2]) {*makespan* da segunda melhor seqüência}

Se $f(s_1) < BM$ então

BS := seq[1]

BM := f(seq[1]).

PASSO 4)

Aplicar a técnica Simulated Annealing tendo s_1 *como solução inicial, com o número de iterações* K = Nvp(S)*, determinando a melhor seqüência* s* *(a de menor makespan).*

$s_1 := s^*$

$f(s_1) := f(s^*)$

Se $f(s_1) < BM$ *então*

$BS := s_1$

$BM := f(s_1)$.

PASSO 5)

Aplicar a técnica Simulated Annealing tendo s_2 *como solução inicial, com o número de iterações* K = Nvp(S)*, determinando a melhor seqüência* s** *(a de menor makespan).*

$s_2 := s^{**}$

$f(s_2) := f(s^{**})$

Se $f(s_2) < BM$ *então*

$BS := s_2$

$BM := f(s_2)$.

PASSO 6)

Tomar as seqüências s_1 *e* s_2 e aplicar o procedimento descrito a partir do *Passo 2*.

- *Resultados:*
  - BS: a melhor seqüência das tarefas (solução do problema);
  - BM: valor do *makespan* ou duração total da programação das tarefas segundo BS.

→ Observação: deve-se notar que no método híbrido HBGASA e nos Algoritmos Genéticos puros, a busca por melhores soluções pode ser encerrada antes da condição de parada dada pelo número total de seqüências TNbS(m,n). Isto ocorre quando as seqüências sobre as quais for aplicado o operador de cruzamento forem iguais (condição de parada secundária).

3. Experimentação Computacional

A experimentação computacional foi dividida em duas etapas:

Na primeira etapa, testou-se o método híbrido Algoritmo Genético-*Simulated Annealing* (HBGASA), utilizando-se o operador de cruzamento de UM CORTE. Foram desenvolvidos 100 procedimentos híbridos distintos, pela combinação das diferentes temperaturas iniciais (T_1) e parcelas da vizinhança analisada (em função do parâmetro p). Tais procedimentos foram comparados com o Algoritmo Genético puro (*1xGA*) – que utilizou o operador de cruzamento de UM CORTE – e com o *Simulated Annealing* puro ModSAfshopH.

Na segunda etapa, testou-se o método HBGASA utilizando-se o operador de cruzamento PMX. Também nessa etapa, foram desenvolvidos 100 procedimentos híbridos, combinando-se as diferentes temperaturas iniciais e parcelas da vizinhança analisada. Esses 100 procedimentos foram comparados com o Algoritmo Genético puro (*pmxGA*) – utilizando o operador de cruzamento PMX – e com o *Simulated Annealing* puro ModSAfshopH.

Em cada etapa foram testados 540 problemas, divididos em 27 classes de acordo com o número de máquinas (m) e o número de tarefas (n), para: m = 4, 7 e 10 máquinas e n = 20, 30, 40, 50, 60, 70, 80, 90 e 100 tarefas. Foram testados 20 problemas em cada classe (m,n).

Nas duas etapas, as temperaturas iniciais (T_1) utilizadas nos algoritmos híbridos foram: T_1 = {20; 30; 45; 60 e 70}. Os valores testados do parâmetro (p) foram: p = {0.05; 0.10; 0.15; 0.20; 0.25; 0.30; 0.35; 0.40; 0.45; 0.50; 0.55; 0.60; 0.65; 0.70; 0.75; 0.80; 0.85; 0.90; 0.95 e 1.00}.

Todos os problemas foram gerados aleatoriamente, sendo os tempos de processamento das tarefas números inteiros distribuídos uniformemente no intervalo [1, 100]. O equipamento utilizado foi um microcomputador Pentium II – Intel MMX de 266 MHz, pertencente ao Laboratório de Multimídia e Processamento Científico da Área de Engenharia de Produção, Escola de Engenharia de São Carlos – Universidade de São Paulo.

Os melhores resultados obtidos referem-se ao método híbrido HBGASA 2030-PMX, ou seja, com temperatura inicial T_1 = 20, parcelas analisadas da vizinhança correspondentes a p = 0,30 e operador de cruzamento PMX. Tais resultados são mostrados nas Figuras 1 a 3, em termos das Porcentagens de Sucesso, Melhorias Relativas e Tempos de Computação, respectivamente, em função do número de tarefas, representando portanto valores médios calculados sobre o conjunto {4, 7, 10} de máquinas.

A *Porcentagem de Sucesso* é calculada pelo número de vezes em que o método obteve a melhor solução, dividido pelo número total de problemas avaliados. Obviamente, quando dois ou mais métodos obtêm a melhor solução para um mesmo problema todos eles alcançam sucesso e conseqüentemente suas porcentagens de sucesso são simultaneamente melhoradas.

A *Melhoria Relativa* mede o percentual de redução da Duração Total da Programação, em relação à solução inicial. Nos métodos que utilizam o Algoritmo Genético, a solução inicial considerada para o cálculo da melhoria relativa é aquela que fornece o menor makespan dentre a solução inicial do FSHOPH e a solução do NEH. É importante ressaltar que todos os métodos comparados entre si têm a mesma solução inicial. A Melhoria Relativa é obtida pela seguinte expressão:

$$RI = (M1 - M) / M1$$ onde:

M1: *makespan* da solução inicial e
M: *makespan* da melhor seqüência que foi encontrada pelo método heurístico.

Hierarquicamente, a estatística que orientou a escolha do melhor método foi a Porcentagem de Sucesso. A Melhoria Relativa é a ela correlacionada e foi usada para reiterar o desempenho quanto à estatística principal. Em outras palavras, quando um método apresenta valores superiores da porcentagem de sucesso, é estatisticamente esperado que a melhoria relativa também apresente valores superiores.

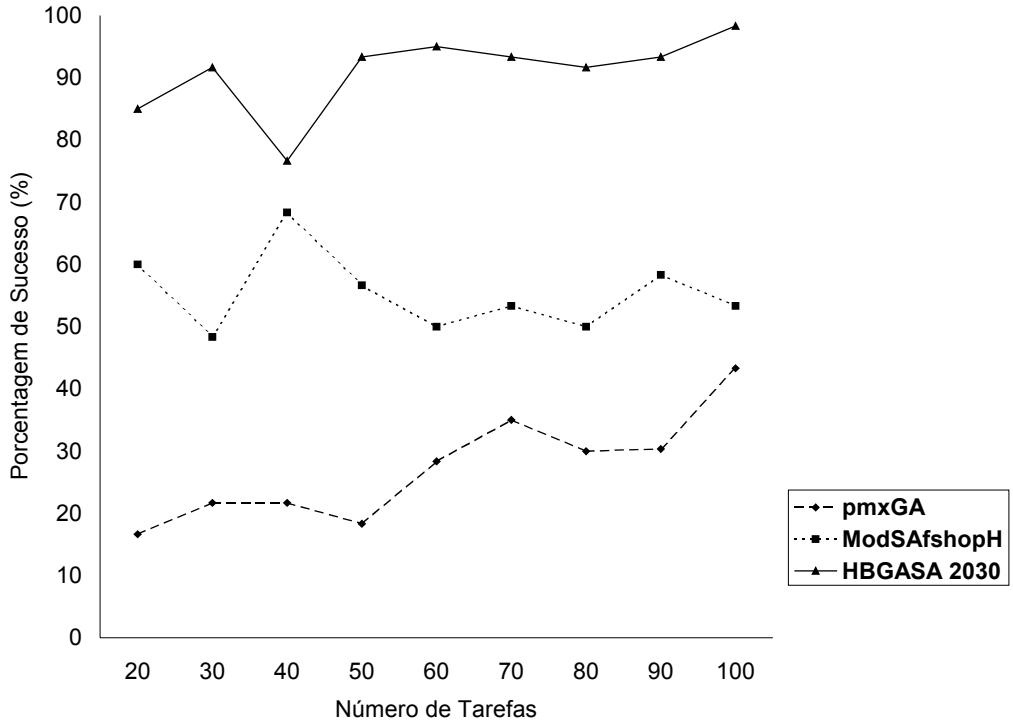

Figura 1 – Porcentagens de Sucesso
pmxGA X ModSAfshopH X HBGASA 2030-PMX.

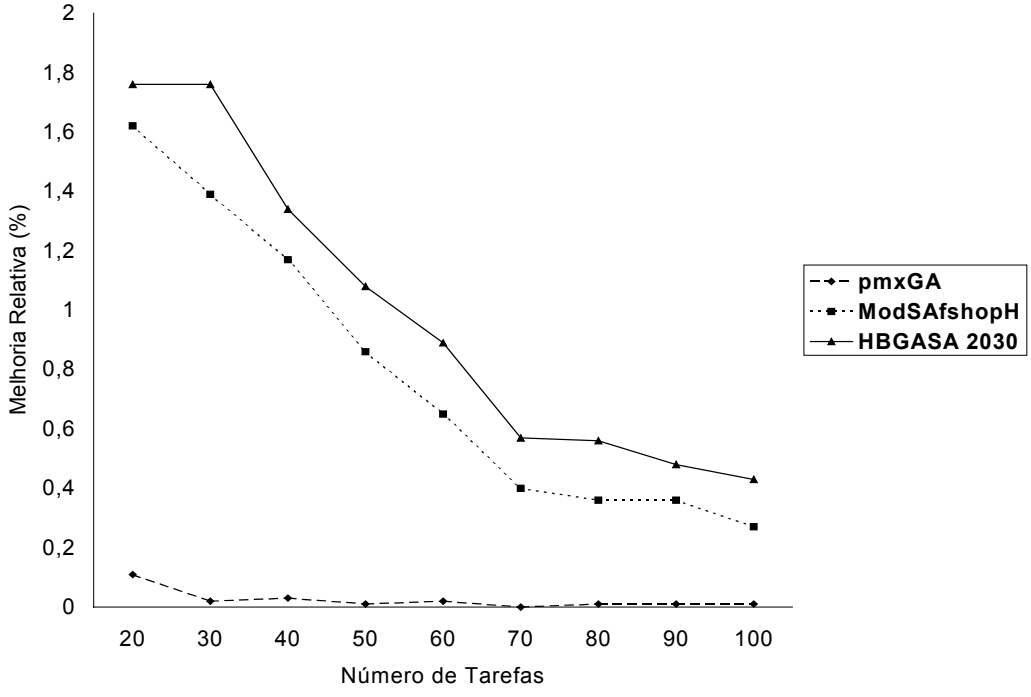

Figura 2 – Melhorias Relativas
pmxGA X ModSAfshopH X HBGASA 2030-PMX.

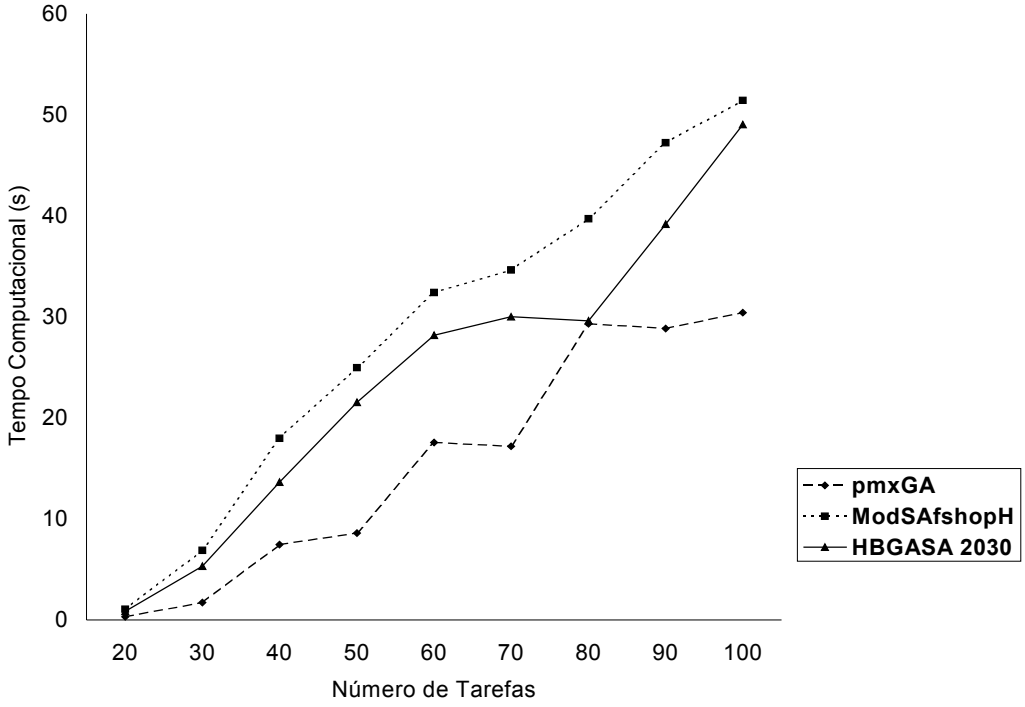

Figura 3 – Tempos de Computação (em segundos) pmxGA X ModSAfshopH X HBGASA 2030-PMX.

Quanto ao *Tempo de Computação*, tendo em vista que os métodos foram projetados para não apresentarem tempos excessivos e grandes diferenças para problemas da mesma classe (m, n), eles são apresentados apenas com o propósito de avaliação dos esforços computacionais.

Os resultados obtidos mostram que o HBGASA 2030-PMX apresenta um desempenho superior em todas as faixas em função do número de tarefas (20 a 100), tanto com relação à Porcentagem de Sucesso quanto em relação à Melhoria Relativa.

De um modo geral, os tempos computacionais demandados pelos métodos ModSAfshopH e HBGASA 2030-PMX foram bem próximos entre si. O esforço computacional menor do algoritmo genético puro pmxGA pode ser explicado pela condição de parada secundária, que encerra a busca de novas soluções quando as seqüências sobre as quais for aplicado o operador de cruzamento forem iguais.

Um aspecto que se salienta diz respeito aos pequenos valores das Melhorias Relativas. Isto é devido ao fato da solução inicial dos métodos ser de boa qualidade (na maioria dos casos a solução do NEH). Melhorias Relativas pequenas quando a solução inicial for boa não significa que o método não é eficaz. Obviamente, se a solução inicial for ótima a melhoria relativa será sempre nula. Uma observação interessante refere-se ao desempenho do Algoritmo Genético puro pmxGA quanto à Melhoria Relativa: os resultados ilustrados na Figura 2 indicam que o operador PMX, quando aplicado de maneira sucessiva e somente combinado com os operadores de mutação, praticamente não conseguiu melhorar a solução inicial.

4. Considerações Finais

A pesquisa relatada neste artigo teve como finalidade a obtenção de um método metaheurístico híbrido para o problema de Programação de Tarefas *Flow Shop* Permutacional, com o objetivo de minimizar o *makespan*. Tal método apresenta as características básicas

do Algoritmo Genético simples (seleção dos pais e aplicação do operador de cruzamento), porém com um importante diferencial: a diversidade genética (que no Algoritmo Genético puro é gerada pelo operador mutação) é obtida pelo método *Simulated Annealing*.

Foram desenvolvidos e testados 200 algoritmos híbridos (denominados HBGASA), combinando-se diferentes operadores de cruzamento, temperaturas iniciais e parcelas da vizinhança analisada. Os algoritmos híbridos foram comparados com Algoritmos Genéticos puros (1xGA / pmxGA) e *Simulated Annealing* puro (ModSAfshopH), os quais foram utilizados em sua concepção, através de ampla experimentação computacional.

Em função dos resultados experimentais obtidos, pode-se dizer que a pesquisa realizada atingiu os seus objetivos (*avaliar a eficácia da hibridização*) e mostrou ser correta a expectativa quanto a métodos metaheurísticos híbridos para a minimização da duração total da programação em ambiente *flow shop* permutacional, ou seja, um método híbrido pode ser desenvolvido de maneira a agregar as características vantajosas dos métodos metaheurísticos puros. Por exemplo, no caso do Algoritmo Genético, a hipótese de que a melhor solução se encontra em regiões do espaço de soluções viáveis que contêm uma grande proporção relativa de boas soluções, e com relação ao *Simulated Annealing* que, assim como a Busca Tabu, tem a capacidade de explorar os "vales" do espaço de soluções na tentativa de obtenção da solução ótima global.

Considerando-se os resultados, observações e conclusões da pesquisa desenvolvida, algumas sugestões para eventuais melhorias no método híbrido HBGASA seriam: o emprego de um operador de cruzamento "inteligente" (não aleatório), que utilize as características genéticas favoráveis (posições relativas das tarefas) nos cromossomos (seqüências) pais; a utilização de diferentes funções de resfriamento e diferentes probabilidades de aceitação, no procedimento *Simulated Annealing*.

Um trabalho futuro que pode ser realizado, refere-se à comparação do HBGASA com os métodos híbridos propostos em MOCCELLIN & SANTOS (2000) e SOUZA & MOCCELLIN (2000), com o objetivo de selecionar o melhor dentre os três métodos que combinam as metaheurísticas Busca Tabu, Algoritmo Genético e *Simulated Annealing*. Tal método poderia, então, ser comparado com o algoritmo de Busca Tabu proposto por NOWICKI & SMUTNICKI (1996), o qual apresenta comprovada eficiência computacional e eficácia quanto à qualidade da solução para o problema de programação da produção tratado neste artigo.

Referências Bibliográficas

DANNENBRING, D.G.: "An Evaluation of Flow-Shop Sequencing Heuristics." *Management Science*, 23: 1174-1182, 1977.

DÍAZ, B.A.: "An SA/TS Mixture Algorithm for the Scheduling Tardiness Problem." *European Journal of Operational Research*, 88: 516-524, 1996.

GAREY, M.R.; JOHNSON, D.S. & SETHI, R.: "The Complexity of Flowshop and Jobshop Scheduling." *Mathematics of Operations Research*, 1: 117-129, 1976.

GLOVER, F.; KELLY, J.P. & LAGUNA, M.: "Genetic Algorithms and Tabu Search: Hybrids for Optimization." *Computers & Operations Research*, 22(1): 111-134, 1995.

IGNALL, E. & SCHRAGE, L.E.: "Application of Branch and Bound Technique to some Flow-Shop Problem." *Operations Research*, 13: 400-412, 1965.

ISHIBUCHI, H.; MISAKI, S. & TANAKA, H.: "Modified Simulated Annealing Algorithms for the Flow Shop Sequencing Problem." *European Journal of Operational Research*, 81: 388-398, 1995.

KIM, H.; NARA, K. & GEN, M.: "A Method for Maintenance Scheduling Using GA Combined with SA." *Computers & Industrial Engineering*, 27: 477-480, 1994.

KURODA, M. & KAWADA, A.: "Improvement on the Computational Efficiency of Inverse Queuing Network Analysis." *Computers & Industrial Engineering*, 27: 421-424, 1994.

MOCCELLIN, J.V.: "Comparison of Neighbourhood Search Heuristics for the Flow Shop Sequencing Problem." *Proceedings of the Fourth International Workshop on Project Management and Scheduling*, 4: 228-231, Leuven-Belgium, July 1994.

MOCCELLIN, J.V.: "A New Heuristic Method for the Permutation Flow Shop Scheduling Problem." *Journal of the Operational Research Society*, 46: 883-886, 1995.

MOCCELLIN, J.V. & NAGANO, M.S.: "Evaluating the Performance of Tabu Search Procedures for Flow Shop Sequencing." *Journal of the Operational Research Society*, 49: 1296-1302, 1998.

MOCCELLIN, J.V. & SANTOS, M.O.: "An Adaptive Hybrid Metaheuristic for Permutation Flowshop Scheduling." *Journal of Control and Cybernetics*, 29: n. 3, 2000.

MOTA, W.L.: *Análise Comparativa de Algoritmos Genéticos para o Problema de Programação de Operações Flow Shop Permutacional*. Dissertação de Mestrado, EESC/USP, 1996.

MURATA, T.; ISHIBUCHI, H. & TANAKA, H.: "Genetic algorithms for flow shop problems." *Computers and Industrial Engineering*, 30(4): 1061-1071, 1996.

NAGANO, M.S.: *Novos procedimentos de Busca Tabu para o problema de programação de operações flow shop permutacional*. Dissertação de Mestrado, EESC/USP, 1995.

NAWAZ, M.; ENSCORE Jr., E.E. & HAM, I.: "A Heuristic Algorithm for the m-Machine, n-Job Flow-Shop Sequencing Problem." *Omega*, 11: 91-95, 1983.

NOWICKI, E. & SMUTNICKI, C.: "A Fast Tabu Search Algorithm for the Permutation Flow-Shop Problem." *European Journal of Operational Research*, 91: 160-175, 1996.

OGBU, F.A. & SMITH, D.K.: "The Application of the Simulated Annealing Algorithm to the Solution of the $n/m/C_{max}$ Flowshop Problem." *Computers and Operations Research*, 17: 243-253, 1990.

OSMAN, I.H. & POTTS, C.N.: "Simulated Annealing for Permutation Flow-Shop Scheduling." *Omega*, 17: 551-557, 1989.

PALMER, D.S.: "Sequencing Jobs through a Multi-stage Process in the Minimum Total Time – A Quick Method of obtaining a Near Optimum." *Operational Research Quarterly*, 16: 101-107, 1965.

PARK, M.W. & KIM, Y.D.: "A Systematic Procedure for Setting Parameters in Simulated Annealing Algorithms." *Computers & Operations Research*, 25: 207-217, 1998.

PIRLOT, M.: "General local search methods." *European Journal of Operational Research*, 92: 493-511, 1996.

REEVES, C.R.: "A Genetic Algorithm for Flowshop Sequencing." *Computers & Operations Research*, 22: 5-13, 1995.

ROACH, A. & NAGI, R.: "A Hybrid GA-SA Algorithm for Just-In-Time Scheduling of Multi-Level Assemblies." *Computers & Industrial Engineering*, 30(4): 1047-1060, 1996.

SELEN, W.J. & HOTT, D.D.: "A Mixed-Integer Goal Programming Formulation of the Standard Flow-Shop Scheduling Problem." *Journal of the Operational Research Society*, 37: 1121-1128, 1986.

SOUZA, A.B.D. & MOCCELLIN, J.V.: "Metaheurística Híbrida Algoritmo Genético-Busca Tabu para Programação de Operações Flow Shop." *Anais do XXXII Simpósio Brasileiro de Pesquisa Operacional*, 32: 314-325, Viçosa-MG, Outubro 2000.

TAILLARD, E.: "Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem." *European Journal of Operational Research*, 47: 65-74, 1990.

WIDMER, M. & HERTZ, A.: "A New Heuristic Method for the Flow Shop Sequencing Problem." *European Journal of Operational Research*, 41: 186-193, 1989.

ZEGORDI, S.H.; ITOH, K. & ENKAWA, T.: "Minimizing Makespan for Flow Shop Scheduling by Combining Simulated Annealing with Sequencing Knowledge." *European Journal of Operational Research*, 85: 515-531, 1995.

PRODUCTION SCHEDULING IN FLOW SHOP SYSTEMS BY USING A HYBRID GENETIC ALGORITHM-SIMULATED ANNEALING HEURISTIC

Abstract

This paper deals with the Permutation Flow Shop Scheduling problem. Many heuristic methods have been proposed for this scheduling problem. A class of such heuristics finds a good solution by improving initial sequences for the jobs through search procedures on the solution space as Genetic Algorithm (GA) and Simulated Annealing (SA). A promising approach for the problem is the formulation of hybrid metaheuristics by combining GA and SA techniques so that the consequent procedure is more effective than either pure GA or SA methods. In this paper we present a hybrid Genetic Algorithm-Simulated Annealing heuristic for the minimal makespan flow shop sequencing problem. In order to evaluate the effectiveness of the hybridization we compare the hybrid heuristic with both pure GA and SA heuristics. Results from computational experience are presented.

Key words: production scheduling, flow shop sequencing, hybrid metaheuristics.