ALGORITMO GENÉTICO COM BUSCA LOCAL PARA A PROGRAMAÇÃO DA PRODUÇÃO EM SISTEMAS FLOW-SHOP HÍBRIDOS

Pedro Luis Miranda Lugo, Rodolfo Florence Teixeira Jr.
Universidade Federal de São Carlos – Departamento de Engenharia de Produção
Rodovia João Leme dos Santos (SP-264), Km 110 - *Sorocaba* - SP - Brasil - CEP 18052-780
pmiranda@ufscar.br, rodolfo.florence@ufscar.br

RESUMO

Nesta pesquisa o problema de *Scheduling* em *Hybrid Flowshop* (*HFS*) com máquinas paralelas não relacionadas, setups dependentes da sequencia e elegibilidade de máquina e *makespan* como critério de otimização é resolvido por um Algoritmo Genético que utiliza um simples operador de busca local. Para atingir uma adequada configuração dos parâmetros do algoritmo proposto, um delineamento experimental *Box-Behnken* é empregado e, baseados em seus resultados, um modelo de regressão é desenvolvido e resolvido com ajuda de software estatísticos e matemáticos. O algoritmo é comparado contra outro Algoritmo Genético que foi especificamente desenvolvido para um problema igual em uma indústria de revestimentos cerâmicos na Espanha. Em total, 1320 instâncias foram utilizadas na avaliação computacional e os resultados alcançados indicam que o algoritmo proposto é melhor tanto em qualidade quanto em esforço computacional para todas as instâncias testadas.

PALAVRAS-CHAVE: *Scheduling, Hybrid Flowshop,* Algoritmo Genético.

ÁREA PRINCIPAL: Administração & Gestão da Produção.

ABSTRACT

In this research, the Hybrid Flowshop (HFS) scheduling problem with unrelated parallel machines, sequence dependent setups times, machine eligibility and makespan as optimization criteria is solved by a Genetic Algorithm, which uses a simple local search operator. To achieve a proper parameters setting of the proposed algorithm, a Box-Behnken experimental design is employed and, based on its results, a regression model is developed and solved with the help of statistical and mathematical software. The algorithm is compared against another Genetic Algorithm that was specifically developed for an equal problem in a ceramic tile industry in Spain. In total, 1320 instances were used in the computational evaluation and the results indicate that the proposed algorithm is better both in quality and computational effort for all instances tested.

KEYWORDS: Scheduling, Hybrid Flowshop, Genetic Algorithm.

MAIN AREA: Administration & Production Management.

1. Introdução

A programação da produção (*scheduling)* é um processo de tomada de decisões em nível operacional, o qual pode ser definido como a alocação dos recursos de produção disponíveis para executar determinadas tarefas de forma eficiente (Jungwattanakit, Reodecha, Chaovalitwongse & Werner, 2009). Adicionalmente, Pinedo (2005) destaca que o *scheduling* é uma função importante dentro de toda organização, pois dela depende que os clientes possam receber os produtos requeridos nas quantidades, momento e lugar indicado. Finalmente, para Herrmann (2006), o *scheduling* permite identificar os recursos em conflito, controlar a liberação dos trabalhos às diferentes estações de processamento, garantir que a matéria-prima requerida seja ordenada na hora oportuna, determinar se as datas de entregas podem ser satisfeitas e definir os tempos disponíveis para realizar a manutenção preventiva.

Neste trabalho, é abordado um problema de *scheduling* em sistemas de produção *Hybrid Flow-Shop (HFS),* o qual é uma generalização do problema clássico *Flow-Shop*, permitindo múltiplas máquinas nas estações de processamento (Yaurima, Burtseva & Tchernykh, 2009). De acordo com Pinedo (2008), o *HFS* consiste em um número de estações em série com um número de máquinas em paralelo em cada uma delas. Aqui, um trabalho pode ser processado em cada estação somente por uma das máquinas disponíveis e, em alguns casos, um trabalho pode "saltar" uma estação se não requer ser processado nela e ir diretamente à frente de outros trabalhos que estão sendo processados, ou à espera de ser, em tal estação.

Este tipo de configuração produtiva pode ser encontrado em diferentes ambientes de manufatura, incluindo fabricação de produtos eletrônicos, indústria de embalagens, setor farmacêutico, fabricação de recipientes de vidro, montagem de automóveis, montagem de placas de circuito impresso (PCB), conformação de metais, entre outros (Chen & Chen, 2009). Outros exemplos incluem a produção de revestimentos cerâmicos (Ruiz & Maroto, 2006) e a inserção de componentes eletrônicos em linhas de montagem de televisão (Yaurima et al., 2009).

O problema em estudo é o *HFS* com máquinas paralelas não relacionadas nas estações de processamento, tempos de setups dependentes da sequencia e elegibilidade de máquinas, visando à minimização do *makespan.* Esta medida de desempenho é de amplo interesse prático porque sua minimização é em certa medida equivalente à maximização da taxa da utilização das máquinas (Pinedo, 2008).

De forma mais formal, seguindo a definição de Ruiz e Maroto (2006), no *HFS* aqui considerado se tem um conjunto N de trabalhos, $N = \{1, 2, \dots, n\}$, que devem ser processados em um conjunto de M estações, $M = \{1, 2, \dots, m\}$. Em cada estação i, $i \in M$, se tem um conjunto $M_i = \{1, 2, \dots, m_i\}$ de máquinas paralelas não relacionadas que podem processar os trabalhos, onde $|M_i| \geq 1$. Cada trabalho deve passar através de todas as estações e deve ser processado por exatamente uma máquina em cada estação. O parâmetro $p_{i,l,j}$ indica o tempo de processamento do trabalho j, $j \in N$, na máquina l, $l \in M_i$, dentro da estação i. Também se tem um tempo de preparação dependente da sequencia baseado na máquina para cada máquina l na estação i quando se processa o trabalho k, $k \in N$, depois de ter processado o trabalho j o qual é denotado como $S_{i,l,j,k}$. Finalmente, para cada trabalho j e estação i se tem um conjunto E_{ij} de máquinas elegíveis que podem processar o trabalho j.

2. Revisão de literatura

Nos últimos anos várias pesquisas envolvendo *HFS* têm sido desenvolvidas. Algumas delas focam-se em problemas mais simples que consideram máquinas paralelas idênticas, número limitado de máquinas nas estações, número de estações limitadas e até tempos de setups independentes, enquanto outras abordam problemas mais complexos que incluem desde fluxos reentrantes até a consideração de múltiplos objetivos.

Choi e Lee (2009) consideram um *HFS* de duas estações com máquinas paralelas e tempos de *setups* independentes da sequencia, visando minimizar o numero de trabalhos atrasados. Li (2007) estuda um *HFS* com máquinas paralelas idênticas em cada estação, considerando tempos de espera limitados entre estações consecutivas, as quais são restrições comuns em indústrias de fabricação de aço e vidro, com o objetivo de minimizar o *makespan*.

Naderi *et al.*(2009) estudam um *HFS* com máquinas paralelas idênticas, tempos de *setups* dependentes da sequência e tempos de transporte, com o objetivo de minimizar o *makespan* ou o *total tardiness.*

Um *HFS* com máquinas paralelas idênticas, tempos de setups dependentes da sequência e tempos de transporte entre estações independentes do trabalho, visando minimizar o *total weighted tardiness* é abordado em (Naderi, Zandieh & Shirazi, 2009). Um *HFS* com máquinas paralelas em cada estação e *buffers* intermédios finitos entre estações consecutivas, com o objetivo de minimizar a soma dos tempos de término ponderados de todos os trabalhos é estudado em (Wang & Tang, 2009).

Um problema que considera tempos indisponíveis para produção foi considerado em (Naderi, Zandieh & Aminnayeri, 2011), onde os autores propõem vários algoritmos para resolver o *HFS* com máquinas paralelas idênticas, incorporando tempos indisponíveis por manutenção preventiva. Um problema de aplicação real é abordado em (Ruiz & Maroto, 2006), onde dados reais de uma indústria de produção de revestimentos cerâmicos são empregados. Um conjunto adicional de instâncias geradas aleatoriamente pelos autores também é analisado. O problema considera máquinas paralelas não relacionadas, tempos de setups dependentes da sequência e elegibilidade de máquinas visando minimizar o *makespan.*

Estudos com múltiplos objetivos têm sido menos abordados e, em consequência, a literatura neste aspecto é relativamente pequena. Behnamian *et al.* (2009) consideram um *HFS* com setups dependentes da sequência, visando minimizar o *makespan* e a soma do *earliness* e *tardiness*. Dugardin *et al.*(2010) estudam um *HFS* reentrante com o objetivo de maximizar a utilização de máquina gargalo e a minimização do *makespan.* Os autores propõem um Algoritmo Genético multi-objetivo que emprega o critério de dominância de Lorenz, em vez do tradicional critério de Pareto, o qual permite atingir um conjunto menor de soluções não dominadas descartando aquelas que são muito boas para um objetivo dado, mas são ruins para o outro objetivo.

Uma revisão ampla e detalhada do *HFS* pode ser consultada em (Ruiz & Vázquez-Rodríguez, 2010), onde mais de 200 artigos são referenciados. O trabalho classifica os artigos estudados de acordo com as variantes analisadas, restrições consideradas, funções objetivo e métodos de solução utilizados.

3. Algoritmo Genético

Nesta seção são descritos os principais operadores e características do Algoritmo Genético proposto como método de solução do problema tratado. Neste algoritmo o tradicional operador genético de mutação é substituído por um operador de Busca Local, como uma tentativa de melhorar a aptidão das soluções na população.

3.1. Codificação e população inicial

Uma permutação de trabalhos tradicional, indicando a ordem na qual os trabalhos serão programados em cada estação, é adotada como esquema de representação das soluções. A regra de alocação de trabalhos à máquina é a ECT (*Earliest Completion Time*), onde um trabalho é alocado à máquina que pode finalizá-lo no tempo mais cedo em uma determinada estação, considerando os diferentes tempos de processamento das máquinas, tempos de setups e elegibilidade das máquinas. Esta regra implica que para cada trabalho na permutação busca-se em todas as estações, procurando a máquina que pode finalizá-lo no tempo mais cedo.

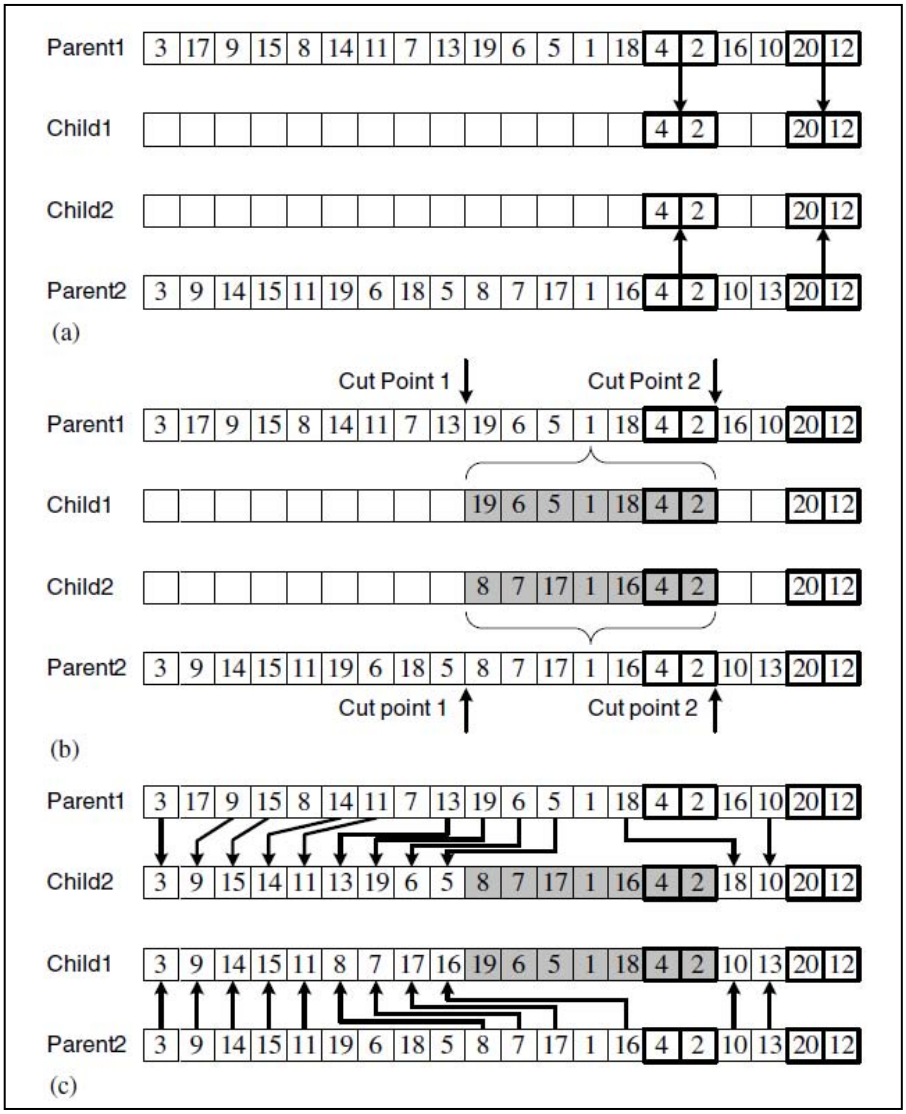
Já a população inicial é formada por um número *N* de indivíduos, ou cromossomos, gerados aleatoriamente.

3.2. Seleção, cruzamento e busca local

Para a seleção foi utilizado o método de Competição Binária Determinística, no qual dois indivíduos são aleatoriamente escolhidos e o melhor deles, em termos de *makespan*, é selecionado como pai.

O cruzamento tem como objetivo a geração de novos indivíduos, denominados filhos, a partir de junção de outros indivíduos, denominados pais, com a esperança de obter soluções com

melhor *makespan*. O operador de cruzamento aqui utilizado é denominado *Similar Block 2-Point Order Crossover (SB2OX)* e foi proposto por Ruiz *et al.* (2005). Na *figura 1* é possível visualizar o esquema do operador de cruzamento *SB2OX*. Inicialmente, os pais são examinados posição por posição. Aqueles grupos com no mínimo dois trabalhos consecutivos idênticos em ambos os pais são denominados blocos, e só eles são diretamente copiado aos filhos (Figura 1(a)). Após isto, dois pontos de corte são gerados aleatoriamente e as seções entre este pontos são copiadas aos filhos (Figura 1(b)). Finalmente, os elementos faltantes em cada filho são copiados em ordem relativo do outro pai (Figura 1(c)).



Fonte: (Ruiz et al., 2005)
Figura 1: Operador *SB2OX*

Como tratado anteriormente, o operador tradicional de mutação dos algoritmos genéticos é substituído por um operador de busca local. De forma geral, este operador gera uma nova solução, denominada solução vizinha, através do intercambio aleatório de duas posições na sequência atual de trabalhos. A nova solução é avaliada e, se o *makespan* for melhor, a sequência atual é substituída pela solução vizinha. O processo é repetido até que o número máximo permitido de vizinhos examinados para a solução atual (*TamVizinhança*) seja alcançado. Este

parâmetro permite manter o tempo de busca na vizinhança dentro limites razoáveis, pois uma busca exaustiva torna-se inviável na medida em que o tamanho do problema aumenta.

3.3. Método de reinicio e esquema de geração dos indivíduos

O método de reinicio é importante para evitar a convergência prematura da população e também que a solução fique presa a um ótimo local (Ruiz et al., 2005). Em cada iteração, o melhor valor *makespan* é armazenado. Se na iteração seguinte esse valor não for alterado, um contador é incrementado. Quando o contador atingir um parâmetro de controle denominado gerações máximas sem melhoramento (*gmsm*), o seguinte procedimento é aplicado:

- Organizar a população em ordem crescente do *makespan;*
- Manter na população os melhores indivíduos baseados no parâmetro *%Mantidos*, o qual indica a percentagem de indivíduos que devem ser selecionados;
- As soluções restantes são descartadas e novamente geradas de forma que a metade é gerada através de mutações do melhor individuo conhecido e a outra metade é gerada aleatoriamente;
- Fazer *Contador=0*.

Este procedimento permite continuar a busca da solução final a partir de um conjunto renovado de indivíduos. Estratégias similares foram utilizadas em Ruiz et al. (2005) e Ruiz e Maroto (2006).

O esquema geracional é o procedimento pelo qual novos indivíduos substituem os indivíduos da iteração anterior. Aqui, um novo indivíduo irá substituir o pior individuo da população se seu *makespan* for melhor e sua sequencia não estiver já contida na população.

4. Configuração experimental do Algoritmo Genético

É sabido que a qualidade das soluções atingidas pelo algoritmo genético é afetada pelos diferentes operadores e níveis dos parâmetros que o compõem. A seguir, é descrita a metodologia utilizada para uma adequada calibração do algoritmo genético proposto, AG_M. A tabela 1 apresenta os diferentes parâmetros e seus respectivos níveis de operação.

Nome	Símbolo	Nível Mínimo	Nível Máximo
Tamanho da população inicial	N	30	50
Número de iterações	*Itermax*	100	200
Probabilidade de Cruzamento	P_c	0.5	0.9
Probabilidade de Busca	P_b	0.2	0.5
Tamanho Vizinhança	*TamVizinhança*	n	$3n$
Gerações máximas sem melhoramento	*gmsm*	5	15
Percentagem de indivíduos mantidos	*%Mantidos*	0.1	0.2

Tabela 1: Parâmetros e níveis de operação do AG_M

Um delineamento experimental de superfície de resposta tipo *Box-Behnken* foi escolhido, pois este tipo de esquema é muito eficiente em termos de corridas experimentais requeridas. Assim, para 7 parâmetros de 3 níveis cada, 62 pontos experimentais foram gerados com a ajuda do Minitab® 16.1.0. Para cada ponto experimental, um subconjunto de 110 problemas foi resolvido, gerando um total de 6820 corridas experimentais. As instâncias empregadas nesta pesquisa são a mesmas usadas em (Ruiz & Maroto, 2006), onde o número de trabalhos n = (20, 50, 100, 200) e o número de estações m = (5, 10, 20). Todas as combinações de n e m são consideradas, exceto 200x5. A tabela 2 resume a características das instâncias.

A primeira coluna mostra o nome do subconjunto. A segunda indica que cada subconjunto tem 110 problemas, todos os quais devem ser resolvidos. Igualmente, a terceira coluna indica a distribuição dos tempos de processamento. Finalmente, a quarta e quinta colunas oferecem informação acerca da distribuição dos tempos de setup e o número de máquinas por estação.

Subconjunto	Número de Instâncias	Tempos de Processamento	Tempos de Setups	Número de Máquinas por estação
SSD10_P13				1 a 3
SSD10_P2	110	*U*[1, 99]	*U*[1, 9]	2
SSD10_P3				3
SSD50_P13				1 a 3
SSD50_P2	110	*U*[1, 99]	*U*[1, 49]	2
SSD50_P3				3
SSD100_P13				1 a 3
SSD100_P2	110	*U*[1, 99]	*U*[1, 99]	2
SSD100_P3				3
SSD125_P13				1 a 3
SSD125_P2	110	*U*[1, 99]	*U*[1, 124]	2
SSD125_P3				3

Tabela 2: Caracterização de instâncias

O subconjunto utilizado na calibração foi o SSD10_P13, conformado por 110 problemas com tempos de processamento, *setups* e máquinas paralelas por estação uniformemente distribuídos entre [1, 99], [1, 9] e [1, 3] respectivamente.

A variável de resposta é:

$$\%Aumento\ sobre\ a\ melhor\ solução = \left(\frac{Heu_{Sol} - Best_{Sol}}{Best_{Sol}} \right) \cdot 100 \qquad (1)$$

Onde Heu_{Sol} é o melhor *makespan* obtido por um determinado algoritmo e $Best_{Sol}$ é o *makespan* obtido para cada instância após de 20000 avaliações de *makespan* do algoritmo genético de Ruiz e Maroto (2006), denominado GA_H, o qual é utilizado como referencial nesta pesquisa.

Seguidamente, um modelo de regressão capaz de explicar a maior variabilidade possível da variável de resposta foi desenvolvido utilizando o software Tinn-R Development Team (2004). A adequação do modelo está condicionada à verificação de alguns pressupostos (normalidade, homogeneidade de variância e independência dos residuais), todos os quais foram validados. Este modelo, relacionando os diferentes parâmetros à variável de resposta, é resolvido usando a ferramenta algébrica *GAMS* fazendo uso dos *solvers* disponíveis dependendo da natureza do modelo. Como resultado final, foram identificados os principais parâmetros do AG_M e seus níveis ótimos de operação. A tabela 3 resume os resultados.

Símbolo	Nível
N	30
Itermax	100
P_c	0.9
P_b	0.223
TamVizinhança	N
gmsm	10
%Mantidos	0.1

Tabela 3: Níveis ótimos de operação do AG_M

5. Avaliação computacional

Após ter calibrado o AG_M foi preciso testar sua eficiência e eficácia. Para isto, foi implementado computacionalmente o algoritmo genético proposto por Ruiz e Maroto (2006), o qual foi originariamente desenvolvido para resolver o problema aqui estudado, permitindo assim uma comparação direta entre algoritmos. Todos os subconjuntos foram avaliados e os resultados serão apresentados ao longo desta seção.

Os algoritmos foram codificados no MATLAB R2009b sob o sistema operacional Windows 7 e, devido ao caráter estocástico dos algoritmos, 5 replicas diferentes para cada subconjunto foram executadas. Todos os testes foram executados em um *notebook* com processador Intel Core i5 com 4 Gb de memória RAM. Os resultados consolidados da experimentação são apresentados nas tabelas 4 e 5.

A tabela 4 atesta o melhor rendimento do AG_M, o qual atingiu percentagens de desvio bem menores que o GA_H. Isto indica o alto nível do algoritmo proposto, o qual partindo de uma população totalmente aleatória é capaz de atingir melhores resultados que o GA_H, onde um membro da população é iniciado a partir do algoritmo NEH, considerado como o melhor algoritmo construtivo para resolver problemas *flow-shop*. Além disso, o AG_M conseguiu em muitos casos melhorar a solução obtida pelo GA_H após 20000 avaliações de *makespan*, como indicam os valores em negrito.

Em geral, o aumento sobre a melhor solução conhecida está na média de 0,08% para o AG_M, enquanto o GA_H tem uma média de 1,61%. O anterior indica claramente a melhor qualidade das soluções encontradas pelo AG_M para todos os subconjuntos testados nesta pesquisa. Também é importante notar que, em geral, o rendimento do AG_M não é significativamente afetado por variações nos valores dos parâmetros (distribuição dos setups e número de máquinas por estação), sendo isto uma evidencia de que a qualidade da solução gerada pelo algoritmo independe de possíveis variações nos parâmetros do problema.

Em relação às instâncias, pode-se notar que o aumento do tamanho destas não tem um impacto significativo sobre a qualidade da solução final do AG_M e, em consequência, o aumento sobre a melhor solução tende a ter pouca variação quando o tamanho da instância aumenta. Por outro lado, o comportamento do GA_H é bem diferente e indica uma melhora significativa da qualidade da solução final na medida em que o tamanho da instância aumenta. No entanto, a evidente melhoria do GA_H não é suficiente para superar o desempenho do AG_M, o qual sempre gera melhores soluções. A *figura 2* ilustra o comportamento do aumento sobre a melhor solução em função do tamanho da instância e permite comprovar a anterior afirmação.

Na tabela 5 pode-se observar o tempo computacional médio de cada algoritmo para os diferentes subconjuntos de experimentação. Os resultados permitem concluir que o AG_M tem uma mínima vantagem sobre o GA_H, com tempos de execução média de 21.49 e 25.88 segundos respectivamente. Da tabela 5 também pode ser observado o comportamento altamente variável do tempo de execução médio do GA_H em aqueles subconjuntos com entre 1 e 3 máquinas por estação, enquanto o AG_M tem um comportamento mais estável, com a média ao redor dos 19 segundos. No entanto, nos demais subconjuntos o GA_H parece estabilizar seu comportamento, conseguindo superar ao AG_M em alguns deles.

Adicionalmente, também pode ser observado que nos subconjuntos com entre 1 e 3 máquinas por estação, o aumento do tempo computacional em função do aumento do tamanho das instâncias é maior para o GA_H. Nos demais subconjuntos o tempo computacional para o GA_H sempre é menor se o tamanho da instância é menor ou igual a 100 x 20. Para instâncias maiores, o tempo computacional do GA_H fica acima do tempo requerido pelo AG_M.

Na *Figura 3* pode se visualizar o crescimento no tempo computacional médio em função do aumento do tamanho das instâncias.

	SSD10_P13		SSD50_P13		SSD100_P13		SSD125_P13		SSD10_P2		SSD50_P2		SSD100_P2		SSD125_P2		SSD10_P3		SSD50_P3		SSD100_P3		SSD125_P3	
Instância	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**
20x5	0,71	0,02	2,83	0,58	4,17	0,43	3,69	**-0,07**	4,26	**-0,58**	4,45	0,41	5,79	0,11	5,22	**-0,74**	5,77	0,56	4,53	**-0,63**	3,35	**-0,86**	5,09	**-0,70**
20x10	1,05	0,05	2,51	1,42	3,22	0,81	3,58	**-0,23**	2,10	**-0,07**	2,51	**-1,21**	3,10	0,16	3,04	**-0,31**	2,43	**-0,34**	3,90	**-0,28**	4,20	**-0,04**	4,18	**-0,91**
20x20	1,96	0,07	2,14	0,71	2,42	**-0,04**	2,89	**-0,04**	2,08	0,19	2,05	**-0,39**	1,89	**-0,77**	2,47	**-0,41**	2,21	0,12	2,33	**-0,14**	2,71	0,29	2,59	**-0,46**
50x5	0,44	**-0,04**	1,13	0,03	1,95	1,20	1,66	0,58	2,00	0,30	2,49	0,66	2,42	0,29	2,05	0,25	2,77	0,14	2,60	0,31	2,47	**-0,17**	3,11	0,31
50x10	0,86	**-0,19**	1,61	0,62	1,41	**-0,58**	1,49	0,99	1,24	0,12	1,79	0,34	1,78	0,34	2,42	0,97	2,76	0,89	1,75	0,17	2,64	0,43	2,34	0,35
50x20	0,83	**-0,06**	0,96	0,30	1,38	**-0,31**	1,53	0,07	1,43	0,23	0,89	0,09	1,50	0,17	1,73	0,13	1,64	0,33	1,90	0,21	1,74	**-0,06**	1,94	0,51
100x5	0,36	**-0,04**	0,43	0,16	0,47	0,74	0,92	**-0,48**	1,13	0,22	1,19	0,64	0,54	**-0,06**	1,67	0,76	1,16	0,05	1,22	0,35	0,87	**-0,06**	1,23	**-0,10**
100x10	0,37	0,00	0,69	**-0,24**	0,70	0,08	0,70	0,24	1,12	0,48	0,81	0,11	0,61	**-0,1**	0,98	0,23	1,01	0,38	1,84	0,93	0,52	**-0,15**	1,17	0,52
100x20	0,40	**-0,16**	0,50	**-0,23**	0,75	**-0,11**	0,45	**-0,46**	0,53	0,14	0,33	**-0,18**	0,42	**-0,13**	0,67	**-0,12**	0,71	0,06	0,65	**-0,05**	0,80	**-0,25**	1,60	1,11
200x10	0,16	**-0,07**	0,06	**-0,12**	0,06	**-0,53**	0,07	0,13	0,24	**-0,02**	0,17	0,01	0,13	0,04	0,03	**-0,07**	0,19	**-0,08**	0,55	0,15	0,47	0,15	0,09	**-0,15**
200x20	0,22	**-0,10**	0,13	**-0,02**	0,06	**-0,03**	0,01	**-0,31**	0,07	**-0,15**	0,03	**-0,14**	0,17	0,02	0,12	**-0,09**	0,27	0,11	0,56	0,29	0,46	0,11	0,54	0,22
Média	0,67	**-0,05**	1,18	**0,29**	1,51	**0,15**	1,54	**0,04**	1,47	**0,08**	1,52	**0,03**	1,67	**0,01**	1,85	**0,05**	1,9	**0,2**	1,98	**0,12**	1,84	**-0,05**	2,17	**0,06**

Tabela 4: Percentagem de aumento médio sobre a melhor solução obtida

	SSD10_P13		SSD50_P13		SSD100_P13		SSD125_P13		SSD10_P2		SSD50_P2		SSD100_P2		SSD125_P2		SSD10_P3		SSD50_P3		SSD100_P3		SSD125_P3	
Instância	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**	**GA_H**	**AG_M**
20x5	3,26	2,86	2,75	2,87	3,07	3,00	2,71	2,96	1,30	2,96	1,33	3,20	1,27	2,93	1,29	3,16	1,54	3,34	1,57	3,39	1,56	3,32	1,42	3,55
20x10	4,84	4,21	3,57	4,16	4,33	4,29	3,55	4,28	1,88	4,29	1,94	4,37	1,85	4,28	1,91	4,60	2,34	5,07	2,32	5,19	2,32	5,18	2,17	5,39
20x20	8,56	6,97	5,59	6,89	7,01	7,17	5,67	7,08	3,06	7,00	3,20	7,11	2,97	6,90	3,15	7,55	3,92	8,48	3,83	8,82	3,86	8,60	3,66	72,50
50x5	8,33	5,95	5,24	5,90	6,28	5,99	5,47	5,81	3,09	6,54	3,09	6,51	2,96	6,13	3,03	6,59	3,82	7,44	3,68	7,61	3,67	7,50	3,45	7,65
50x10	12,60	9,73	7,51	9,33	10,45	9,62	7,59	9,21	4,66	9,73	4,64	10,00	4,46	9,52	4,70	10,05	5,90	11,95	5,74	12,31	5,85	12,26	5,64	12,70
50x20	21,03	16,76	11,72	15,77	16,83	16,29	12,67	15,78	7,69	16,10	7,80	16,78	7,53	16,08	7,84	16,99	10,18	20,89	9,91	21,40	10,20	21,56	10,22	22,16
100x5	23,41	12,36	12,04	11,91	18,34	12,24	12,52	11,67	8,07	11,94	7,98	12,29	7,66	11,37	7,80	12,10	9,73	14,06	9,73	14,26	9,81	14,05	9,57	14,07
100x10	34,85	20,15	17,60	18,67	27,68	19,01	19,14	18,15	12,40	18,58	12,36	18,34	11,96	18,01	12,46	18,80	15,82	22,91	15,70	23,44	15,88	23,33	15,43	24,49
100x20	58,62	34,57	30,81	31,68	47,21	32,16	31,90	31,38	21,16	31,40	21,57	31,22	20,71	30,99	21,46	32,74	28,03	41,10	27,65	41,69	28,78	41,68	27,64	44,01
200x10	147,54	39,16	72,74	37,18	96,34	37,59	71,52	36,38	52,21	36,17	52,68	36,51	50,13	35,78	52,40	37,24	67,66	45,53	66,43	47,02	68,44	47,71	65,76	47,68
200x20	225,01	67,94	127,15	65,40	126,24	65,93	134,05	64,22	91,44	64,60	92,68	62,65	88,57	63,34	91,67	63,64	125,06	83,16	120,16	84,91	123,34	88,15	117,96	87,69
Média	49,82	**20,06**	26,98	**19,07**	33,07	**19,39**	27,89	**18,81**	18,81	19,03	19,03	**19,00**	18,19	18,67	18,88	19,40	24,91	**23,99**	24,24	24,55	24,88	**24,85**	23,90	31,08

Tabela 5: Tempo computacional médio (segundos)

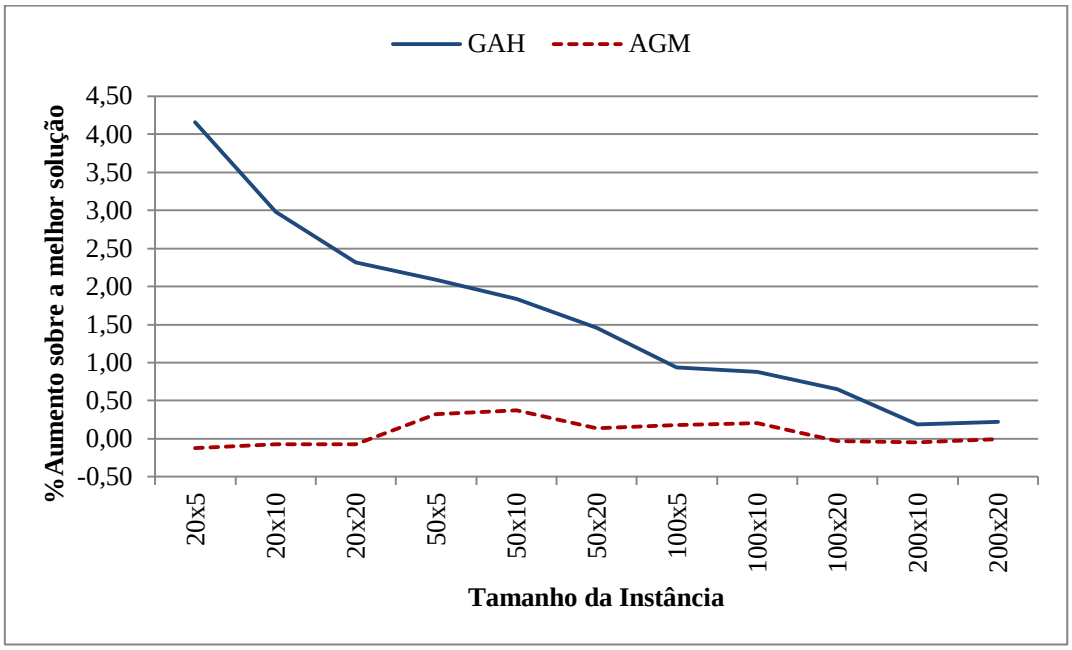


Figura 2: Percentagem de aumento sobre a melhor solução em função do tamanho da instância

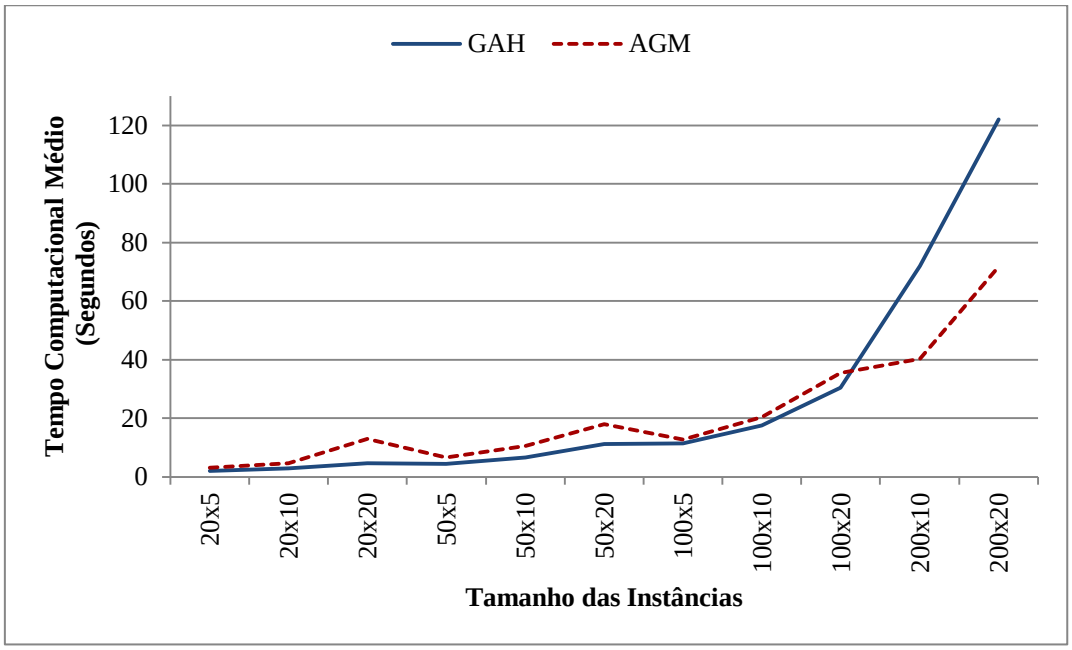


Figura 3: Tempo computacional médio em função do tamanho das instâncias

6. Conclusões e trabalhos futuros

Nesta pesquisa foi proposto um algoritmo genético AG_M, com o objetivo de resolver o problema *HFS* com máquinas paralelas não relacionadas, setups dependentes da sequência e elegibilidade de máquinas. A configuração dos parâmetros do AG_M foi feita através da execução de um delineamento experimental tipo *Box-Behnken*, a construção de um modelo de regressão derivado dos resultados do delineamento e, finalmente, a resolução do modelo utilizando *solvers* disponíveis na ferramenta *GAMS*. O AG_M foi comparado com o algoritmo genético GA_H, proposto na literatura para a resolução do mesmo problema. Um amplo conjunto de instâncias foi testado com o objetivo de analisar o comportamento do algoritmo proposto para diferentes tipos e

tamanhos de problemas. Os resultados indicam claramente a superioridade do AG_M para todas as instâncias testadas.

Mais especificamente, os resultados mostraram que o AG_M pode resolver instâncias de pequeno, médio e grande porte com um aumento médio sobre a melhor solução conhecida de 0,08%, enquanto o GA_H atinge soluções na média de 1,61%. Igualmente, o AG_M mostrou não ser afetado consideravelmente por possíveis variações nos parâmetros do problema como distribuição dos setups, número de máquinas por estação e tamanho da instância, garantindo sempre soluções de boa qualidade para todos os problemas resolvidos.

Em relação ao tempo computacional, ambos os algoritmos tem comportamentos similares. Quando comparados sobre a base dos tamanhos das instâncias, o GA_H precisa menos tempo se a instância não é maior a 100 x 20. Para instâncias maiores o AG_M supera claramente, precisando de entre 30 e 50 segundos a menos para resolver instâncias de 200 x 10 e 200 x 20 respectivamente.

Como pesquisas futuras, pretende-se empregar instâncias mais complexas com um maior número de máquinas paralelas em cada estação, considerando também tempos de transporte, buffers limitados e múltiplos objetivos, visando aproximar mais o problema estudado com a realidade da indústria.

7. Agradecimentos

Os autores agradecem ao Dr. Rubén Ruiz García do Grupo de Sistemas de Otimização Aplicada (SOA) do Instituto Tecnológico de Informática (ITI) da Universidade Politênica de Valencia (Espanha), que forneceu todas as instâncias e parte do código de seu algoritmo genético para ser usado ao longo desta pesquisa.

8. Referências

Behnamian, J., Fatemi Ghomi, S., & Zandieh, M. (2009). A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. *Expert Systems with Applications*, *36*(8), 11057-11069. Elsevier Ltd. doi:10.1016/j.eswa.2009.02.080

Chen, C., & Chen, C. (2009). A bottleneck-based heuristic for minimizing makespan in a flexible flow line with unrelated parallel machines. *Computers & Operations Research*, *36*(11), 3073-3081. doi:10.1016/j.cor.2009.02.004

Choi, H.-S., & Lee, D.-H. (2009). Scheduling algorithms to minimize the number of tardy jobs in two-stage hybrid flow shops. *Computers & Industrial Engineering*, *56*(1), 113-120. Elsevier Ltd. doi:10.1016/j.cie.2008.04.005

Dugardin, F., Yalaoui, F., & Amodeo, L. (2010). New multi-objective method to solve reentrant hybrid flow shop scheduling problem. *European Journal of Operational Research*, *203*(1), 22-31. Elsevier B.V. doi:10.1016/j.ejor.2009.06.031

Herrmann, J. W. (2006). A History of Production Scheduling. *Handbook of Production Scheduling* (p. 320). New York.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers & Operations Research*, *36*(2), 358-378. doi:10.1016/j.cor.2007.10.004

Li, T. (2007). Constructive Backtracking Heuristic for Hybrid Flowshop Scheduling with Limited Waiting Times. *Science And Technology*, 6671-6674.

Naderi, B., Zandieh, M., & Aminnayeri, M. (2011). Incorporating periodic preventive maintenance into flexible flowshop scheduling problems. *Applied Soft Computing*, *11*(2), 2094-2101. doi:10.1016/j.asoc.2010.07.008

Naderi, B., Zandieh, M., & Shirazi, A. (2009). Modeling and scheduling a case of flexible flowshops: Total weighted tardiness minimization, *57*, 1258-1267. doi:10.1016/j.cie.2009.06.005

Naderi, B., Zandieh, M., Khaleghi Ghoshe Balagh, A., & Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert Systems with Applications*, *36*(6), 9625-9633. Elsevier Ltd. doi:10.1016/j.eswa.2008.09.063

Pinedo, M. L. (2005). *Planning and Scheduling in Manufacturing and Services*. *New York* (p. 506). New York: Springer.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. *American Society of Mechanical Engineers* (3rd ed., p. 671). New York: Springer.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, *169*(3), 781-800. doi:10.1016/j.ejor.2004.06.038

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, *205*(1), 1-18. doi:10.1016/j.ejor.2009.09.024

Ruiz, R., Maroto, C., & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research*, *165*(1), 34-54. doi:10.1016/j.ejor.2004.01.022

Wang, X., & Tang, L. (2009). A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers. *Computers & Operations Research*, *36*(3), 907-918. doi:10.1016/j.cor.2007.11.004

Yaurima, V., Burtseva, L., & Tchernykh, A. (2009). Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. *Computers & Industrial Engineering*, *56*(4), 1452-1463. Elsevier Ltd. doi:10.1016/j.cie.2008.09.004