Heurística Evolutiva para a Redução do Estoque em Processamento em Sistemas de Produção Flow Shop Permutacional

Marcelo Seido Nagano †
Geraldo Ribeiro Filho ‡
Luiz Antonio Nogueira Lorena §

† Escola de Engenharia de São Carlos
Universidade de São Paulo
São Carlos, São Paulo - Brasil
drnagano@usp.br

‡ Faculdade Bandeirantes de Educação Superior
Suzano, São Paulo - Brasil
geraldorf@uol.com.br

§ Instituto Nacional de Pesquisas Espaciais
São José dos Campos, São Paulo – Brasil
lorena@lac.inpe.br

Abstract

This paper deals with the Permutation Flow Shop scheduling problem with the objective of minimizing total flow time, therefore reducing in-process inventory. An evolutionary heuristic is proposed for the scheduling problem solution. The proposed method is compared with the bests results reported in the literature. Experimental results show that the new method provides better solutions regarding the solution quality.

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. Uma heurística evolutiva é proposta para a solução do problema. Por meio de experimentação computacional, o desempenho do método proposto é avaliado e comparado com os melhores resultados reportados na literatura. Os resultados experimentais mostraram a superioridade do novo método para o conjunto de problemas tratados em relação à qualidade das soluções obtidas.

Keywords: Permutation Flow shop, In-process inventory, Metaheuristic, Genetic algorithm.

Title: Evolutionary clustering search for flowtime minimization in permutation flow shop.

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!*.

O problema consiste em obter uma seqüência das tarefas que otimiza uma determinada medida de desempenho. Nos modelos para solução do problema, as medidas usuais referem-se à minimização da duração total da programação (*makespan*), associada à utilização eficiente dos recursos produtivos, e à minimização da soma dos tempos de fluxo das tarefas (*total flow time*) associados à redução do estoque em processamento, sendo esta última a medida adotada neste trabalho.

Este problema de programação da produção é NP-hard (Garey et al., 1976 e Rinnooy Kan, 1976), e portanto, a busca por uma solução ótima apresenta importância mais teórica do que prática, direcionando as pesquisas para o desenvolvimento de métodos heurísticos e metaheuristicos, sendo os mais recentes descritos sucintamente a seguir.

2 Métodos Heurísticos para Minimização da Soma dos Tempos de Fluxo das Tarefas

Atualmente na literatura os métodos heurísticos podem ser divididos em duas grandes classes: os métodos heurísticos construtivos e os melhorativos.

Na literatura os primeiros métodos heurísticos construtivos para o problema foram as heurísticas propostas por Gupta (1972), Miyazaki et al. (1978), Rajendran e Chaudhuri (1991) e Rajendran (1993). Rajendran e Chaudhuri desenvolveram três heurísticas que obtiveram melhor desempenho que os métodos heurísticos propostos por Gupta e Miyazaki, tanto em termos da qualidade de solução como também quanto ao esforço computacional.

Ho (1995) propôs uma heurística composta de diferentes iterações no processo de melhoria de uma solução inicial, a partir da obtenção de um ótimo local por permutação de pares de tarefas adjacentes, melhorando a solução com movimentos de inserção de tarefas. Tal método heurístico apresentou desempenho melhor que os anteriores reportados na literatura, embora, como esperado, com desvantagem quanto ao tempo de computação, pelo fato de não ser uma heurística construtiva, apresentando semelhanças com metaheurísticas como Simulated Annealing e Busca Tabu.

Rajendran e Ziegler (1997) propuseram a heurística *RZ*, que consiste de duas fases: na primeira, uma seqüência inicial é gerada utilizando uma regra de prioridade similar ao *shortest weighted total processing time*; na segunda fase, a seqüência inicial obtida é melhorada por meio de inserções das tarefas em seqüências parciais, sucessivamente obtidas.

Wang et al. (1997) desenvolveram duas heurísticas denominadas *LIT* (*less idle time*) e *SPD* (*smallest process distance*). Suas heurísticas não são comparadas com as existentes, mas sim com um limitante inferior do tempo médio de fluxo, proposto por Ahmadi e Bagchi (1990).

Woo e Yim (1998) desenvolveram uma heurística denominada *WY*, comparando seu desempenho com a heurística de Rajendran (1993), e também com adaptações do *NEH*

(Nawaz et al., 1983) e *CDS* (Campbell et al., 1970) para o critério de minimização do tempo médio de fluxo. Eles verificaram que seu método apresentava desempenho superior aos outros, principalmente comparado com Rajendran.

Liu e Reeves (2001) apresentaram uma heurística flexível construtiva *H(x)*, que produzia *x* programas *(x=1,2,..n)*. O procedimento heurístico construía uma solução *S* adicionando tarefas uma de cada vez obtidas de um conjunto *U* de tarefas não programas. As tarefas em *U* eram classificadas de acordo com uma função de índice que dependia do número de tarefas já seqüenciadas. A função de índice era uma combinação ponderada entre o tempo total de máquina parada e o tempo total de fluxo.

Allahverdi e Aldowaisan (2002) apresentaram sete heurísticas denominadas *IHx (x=1,2,..,7)*. O melhor desempenho foi alcançado pela heurística *IH7*, constituída de três fases. Na primeira fase, uma solução inicial é obtida aplicando-se a heurística *WY* (Woo e Yim, 1998). Esta primeira solução é utilizada na segunda fase como seqüência inicial para a heurística *RZ* (Rajendran e Ziegler, 1997). Finalmente, a segunda solução é melhorada por meio de uma busca local, com procedimentos de permutação nas posições das tarefas.

Framinan e Leisten (2003) propuseram o método denominado *FL-IH7*, inicialmente denominado *IH7-proposed*, que consiste em uma extensão do *IH7*. A diferença relevante situa-se na obtenção da solução inicial do processo de três fases do *IH7*. No *FL-IH7* tal solução inicial é obtida por um método construtivo que utiliza inicialmente o mesmo procedimento de obtenção de seqüências parciais do *NEH* (Nawaz et al., 1983), porém, ordenando as tarefas de acordo com a soma dos tempos de processamento não-decrescentes. A seguir, as seqüências parciais são melhoradas por um procedimento de busca completa nas respectivas vizinhanças de permutação. Por meio de uma experimentação computacional constatou-se que o *FL-IH7* apresenta um desempenho melhor quando comparado com o *IH7* original.

Framinan et al. (2005) apresentaram uma extensão da pesquisa realizada por Framinan e Leisten (2003). Na pesquisa as heurísticas *proposed* e *IH7-proposed* são respectivamente renomeadas para *FL* e *IH7-FL*. Os autores apresentam duas novas heurísticas compostas, nomeadas de *C1* e *C2*. As novas heurísticas *C1* e *C2* são comparadas com as heurísticas *FL* e *IH7-FL* e também com a original heurística *IH7* de Allahverdi e Aldowaisan (2002). Os resultados da experimentação computacional mostraram que as heurísticas existentes apresentam desempenho inferior que a heurística *C2*.

Li et al. (2009) propuseram uma nova heurística com a aceleração do cálculo do tempo total de fluxo, onde uma nova seqüência gerada é resultante da composição de duas subseqüências. Três heurísticas compostas são apresentadas (*ICH1*, *ICH2* e *ICH3*), e os resultados computacionais mostraram que a heurística *ICH3* apresentou os melhores resultados, além de maior eficiência computacional, quando comparada a outros métodos existentes na literatura.

Mais recentemente, Nagano e Moccellin (2008) apresentaram um novo método heurístico denominado *NM-flowtime*. A heurística proposta é composta de três fases. Na primeira as tarefas são ordenadas de acordo com os valores não-decrescentes das somas dos tempos de processamento para cada tarefa em todas as máquinas. A segunda fase utiliza-se o método de inserção de tarefas semelhante ao método *NEH* (Nawaz et al., 1983) para obtenção da solução inicial. Finalmente, a solução inicial é melhorada utilizando procedimentos de busca local baseado em vizinhanças de permutação e inserção de tarefas para as parciais seqüências obtidas.

Em relação aos métodos melhorativos, Rajendran e Ziegler (2004) desenvolveram dois métodos metaheurísticos baseado no algoritmo de otimização de colônia de formigas (*Ant-*

colony algorithm (Stuetzle, 1998)). O primeiro método estende a idéia do algoritmo de colônia de formigas, chamado de *MMAS* (*max-min ant system*), incorpora a sugestão proposta por Merkle e Middendorf (2000) e uma nova proposta de busca local (*M-MMAS*). O segundo método é chamado de *PACO* (*proposed ant-colony algorithm*). Os métodos foram avaliados com os 90 problemas (*benchmark*) de Taillard (1993). Os resultados obtidos mostraram que os dois métodos de colônia de formigas obtiveram melhores resultados em média, que o *MMAS* para o critério de minimização do *makespan*.

Considerando o critério de minimização da soma total dos tempos de fluxos das tarefas em relação às melhores soluções dos problemas de Taillard (1993) apresentados por Liu e Reeves (2001), os algoritmos de colônia de formigas denominados de *M-MMAS* e *PACO* obtiveram melhores resultados, mostrando serem claramente superiores em comparação as heurísticas de Liu e Reeves e também em relação as melhores soluções reportadas em seu trabalho.

Recentemente, Tasgetiren et al., (2007) apresentaram uma metaheurística *PSO* (*particle swarm optimization*) para os mesmos problemas apresentados por Rajendran e Ziegler (2004). Em sua pesquisa uma regra heurística chamada *SPV* (*smallest position value*) proposto por Bean (1994) foi desenvolvida para habilitar o *PSO* à aplicação aos problemas de sequencimento. Adicionalmente, uma eficiente busca local, chamada de *VNS* (*variable neighborhood search*) é incorporada ao *PSO*. O método foi testado para os problemas (*benchmark*) de Taillard (1993) e também para os problemas (*benchmark*) Watson et al. (2002). Para o critério de minimização da soma total dos tempos de fluxos verifica-se que as soluções obtidas são melhores quando comparadas a Liu e Reeves (2001) e Rajendran e Ziegler (2004).

De acordo com a revisão da literatura efetuada na pesquisa relatada neste trabalho, o método PSO_{VNS} proposto por Tasgetiren et al. (2007) é o melhor método metaheurístico em termos de qualidade da solução para a Programação Flow Shop Permutacional com o critério de minimização da soma total dos tempos de fluxo da tarefa.

3. Heurística Evolutiva para o Problema de Flow Shop Permutacional

Neste trabalho foi utilizada uma heurística evolutiva (HE) com base nos clássicos Algoritmos Genéticos (AG), aplicada ao problema de programação Flow Shop Permutacional.

A representação usada na HE para os cromossomos foi um vetor com n posições, indicando diretamente a seqüência de tarefas da solução. O tamanho da população da HE foi definido como fixo, com 500 indivíduos.

Como a qualidade dos indivíduos da população inicial é de grande importância nos resultados dos AG, para garantir esta qualidade, no processo de inicialização da população foi utilizada a variação da heurística de *NEH* (Nawaz et al. (1983)). Na heurística *NEH* original, inicialmente um conjunto de n tarefas é ordenado de acordo com os valores não-decrescentes da soma dos tempos de processamento em todas as máquinas, em seguida, as duas primeiras tarefas da ordenação são seqüenciadas de modo a diminuir a soma do tempo total de fluxo desta seqüência parcial, as demais tarefas, a partir da terceira, são então inseridas (uma a uma) em uma posição da seqüência parcial que leve à menor soma total do tempo de fluxo das tarefas. Na variação usada neste trabalho, após a ordenação, as duas primeiras tarefas a serem seqüenciadas são definidas aleatoriamente entre as n tarefas do problema.

O primeiro indivíduo inserido na população inicial foi gerado pelo procedimento *NEH* original, em seguida um número de indivíduos dado por *Min {(n*(n-1)/2)/2, 0,5*500}* foram gerados pela variação do *NEH* acima descrita.

Portanto, no máximo 50% dos indivíduos da população inicial foram produzidos com a heurística *NEH* e sua variação. Com o objetivo de garantir diversidade na população inicial, os indivíduos restantes foram gerados com permutações aleatórias das *n* tarefas.

A avaliação dos indivíduos foi feita diretamente pelo critério de minimização da soma do tempo total de fluxo das tarefas, e a rotina de inserção de indivíduos manteve a população ordenada, com o melhor indivíduo (aquele com a menor soma total do tempo de fluxo) ocupando a primeira posição na população. Esta mesma rotina de inserção também não permitia a inserção de indivíduos repetidos na população.

Como critério de parada do processo evolutivo foi adotado o número máximo de 100 iterações ou o número de 20 iterações sucessivas sem que nenhum novo indivíduo fosse inserido na população. O número de tentativas de inserção de novos indivíduos a cada iteração foi igual a 50 (10% do tamanho da população).

A cada tentativa de gerar um novo indivíduo dois pais eram selecionados na população, um deles entre os melhores 40% da população, chamado de indivíduo base, e outro entre todos da população, chamado de indivíduo guia. Um processo de cruzamento, ou recombinação, conhecido *BOX* (*Block Order Crossover*) (Syswerda, 1989), foi então usado para gerar os novos indivíduos. Nesta técnica, os indivíduos pais são combinados através da cópia aleatória de blocos de genes de ambos os pais, o que resulta na geração de um único filho contendo a carga genética dos pais. Neste trabalho, o filho foi gerado com 50% dos genes de cada pai. A figura 1 ilustra o esquema de recombinação BOX.

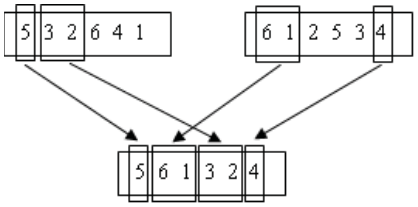

Figura 1: Recombinação BOX

Após a recombinação, o novo indivíduo tinha 60% de probabilidade de passar por um processo de melhoria na forma de uma busca local *LS1*, conforme apresentada na figura 2.

```
Busca_LS1(Solução_inicial)
{
  sa = Solução_inicial;  // solução atual
  Fim = Falso;
  Enquanto (Fim == Falso) {
    vp = Vizinhança_permutação(sa);
    sp = Varre vp até que sp seja melhor que sa
         ou sp seja o melhor em vp;
    vi = Vizinhança_inserção(sa);
    si = Varre vi até que si seja melhor que sa
         ou si seja o melhor em vi;
    Se (sp é melhor que si e sa)
      sa = sp;
    Senão
      Se (si é melhor que sp e sa)
        sa = si;
      Senão
        Fim = Verdadeiro;
  }
  Retorna(sa);
}
```

Figura 2: Busca local de melhoria dos indivíduos

Esta busca por uma solução melhorada foi feita por um processo híbrido que utiliza dois tipos de vizinhança (de permutação e de inserção). Na vizinhança de permutação todos os possíveis pares de genes do cromossomo são trocados, gerando assim $n*(n-1)/2$ novos cromossomos. Na vizinhança de inserção, cada gene é removido de sua posição e inserido em todas as possíveis posições entre os demais genes, deslocando-os para preencher a posição deixada por ele, gerando assim $n*(n-1)$ novos cromossomos.

Cada novo indivíduo, melhorado pela busca *LS1* na maioria das vezes, e que não fosse idêntico a nenhum indivíduo já pertencente à população, era inserido na população em uma posição referente à sua avaliação, provocando assim a remoção do pior indivíduo presente na população até aquele momento. Portanto, o processo evolutivo eventualmente atualizava a população a cada novo indivíduo gerado.

No final das iterações, o primeiro indivíduo da população era a melhor solução obtida até aquele momento.

4. Experimentação Computacional e Métodos de Análise

Para a avaliação do desempenho do método proposto, foram realizados testes utilizando os problemas de Taillard (1993) com n igual 20, 50 e 100 tarefas e m igual 5, 10 e 20 máquinas, conforme utilizados por Liu e Reeves (2001), Rajendran e Ziegler (2004), Li et al. (2009) e Tasgetiren et al. (2007).

Para este trabalho, o algoritmo foi codificado em linguagem C e processado em um microcomputador Pentium IV de 3.0 GHz, com 1 GByte de RAM.

As estatísticas usadas para avaliar o desempenho do método foram a Porcentagem de Sucesso e o Desvio Relativo Médio. A primeira é definida pelo quociente entre o número total de problemas para os quais o método obteve a melhor soma total dos tempos de fluxo das tarefas, e o número total de problemas resolvidos. A segunda quantifica o desvio relativo (DR_h) que o método h obtém em relação à melhor soma dos tempos de fluxo das tarefas obtido para um mesmo problema, sendo calculado por:

$$DR_h = \frac{(F_h - F_*)}{F_*} \quad (1)$$

Onde:

F_h : soma dos tempos de fluxo das tarefas obtido pelo método h;
F_* : melhor soma dos tempos de fluxo das tarefas obtido pelos métodos para um determinado problema.

5. Análise dos Resultados

Para avaliação do desempenho, os resultados da heurística evolutiva HE foram comparados com dois algoritmos baseados em colônias de formigas (*M-MMAS* e *PACO*), apresentados por Rajendran e Ziegler (2004), e um método baseado em enxame de partículas (*PSO*) apresentado por Tasgetiren et al. (2007). Os resultados principais são apresentados nas tabelas 1 e 2.

A tabela 1 apresenta a melhor solução obtida com os métodos. Verifica-se assim a superioridade do novo método HE para o conjunto de problemas avaliados. Para o total de 90 problemas HE apresentou a melhor solução em 79 problemas o que corresponde a 87,8% do total de problemas, sendo que 57 soluções são inéditas (63,3%).

A tabela 2 apresenta a porcentagem de sucesso por classes de problemas. Pode-se verificar que o método HE apresentou porcentagens de sucesso entre 60% e 100%. A tabela 2 mostra ainda que a diferença entre os desvios relativos médios do método HE e os desvios dos outros métodos é bem acentuada, consubstanciando a superioridade do método proposto, já mostrada pelas porcentagens de sucesso.

Tabela 1: Melhores soluções obtidas para os problemas de Taillard

n	*m*	*M-MMAS*	*PACO*	PSO_{vns}	HE
20	5	14056	14056	14033	14033
		15151	15214	15151	15151
		13416	13403	13301	13301
		15486	15505	15447	15447
		13529	13529	13529	13529
		13139	13123	13123	13123
		13559	13674	13548	13548
		13968	14042	13948	13948
		14317	14383	14295	14295
		12968	13021	12943	12943
20	10	20980	20958	20911	20911
		22440	22591	22440	22440
		19833	19968	19833	19833
		18724	18769	18710	18710
		18644	18749	18641	18641
		19245	19245	19249	19245
		18376	18377	18363	18363
		20241	20377	20241	20241
		20330	20330	20330	20330
		21320	21323	21320	21320
20	20	33623	33623	34975	33623
		31604	31597	32659	31587
		33920	34130	34594	33920
		31698	31753	32716	31661
		34593	34642	35455	34557
		32637	32594	33530	32564
		33038	32922	33733	32922
		32444	32533	33008	32412

		33625	33623	34446	33600
		32317	32317	33281	32262
50	5	65768	65546	65058	64924
		68828	68485	68298	68239
		64166	64149	63577	63500
		69113	69359	68571	68469
		70331	70154	69698	69599
		67563	67664	67138	67041
		67014	66600	66338	66411
		64863	65123	64638	64565
		63735	63483	63227	63068
		70256	69831	69195	69144
50	10	89599	88942	88031	87683
		83612	84549	83624	83535
		81655	81338	80609	80493
		87924	88014	87053	86934
		88826	87801	87263	86893
		88394	88269	87255	87027
		90686	89984	89259	89304
		88595	88281	87192	87316
		86975	86995	86102	86280
		89470	89238	88631	88534
50	20	127348	126962	128622	126315
		121208	121098	122173	119502
		118051	117524	118719	116910
		123061	122807	123028	121031
		119920	119221	121202	118914
		122369	122262	123217	121087
		125609	125351	125586	123340
		124543	124374	125714	123005
		124059	123646	124932	122428
		126582	125767	126311	124785
100	5	257025	257886	254762	255023
		246612	246326	245315	243943
		240537	241271	239777	239002
		230480	230376	228872	228928
		243013	243457	242245	241659
		236225	236409	234082	234172
		243935	243854	242122	241753
		234813	234579	232755	232315
		252384	253325	249959	249680
		246261	246750	244275	244287
100	10	305004	305376	303142	301176
		279094	278921	277109	277082
		297177	294239	292465	290994
		306994	306739	304676	304427
		290493	289676	288242	287657
		276449	275932	272790	273065
		286545	284846	282440	282467
		297454	297400	293572	294119
		309664	307043	305605	304964
		296869	297182	295173	294446
100	20	373756	372630	374351	371391
		383614	381124	379792	376383
		380112	379135	378174	374599
		380201	380765	380899	378550
		377268	379064	376187	374429
		381510	380464	379248	377567
		381963	382015	380912	378367
		393617	393075	392315	389680
		385478	380359	382212	380152
		387948	388060	386013	383928

Tabela 2: Porcentagem de sucesso [a] e Desvio relativo médio (%) [b]

n	*m*	*M-MMAS*	*PACO*	PSO_{vns}	HE
20	5	20 [a]	20	100	100
		0.1975 [b]	0,4544	0,0000	0,0000
20	10	60	20	90	100
		0,0492	0,3235	0,0021	0,0000
20	20	20	20	0	100
		0,1195	0,1892	2,8278	0,0000
50	5	0	0	10	90
		1,1302	0,9450	0,2452	0,0110
50	10	0	0	30	70
		1,4196	1,1569	0,1841	0,0248
50	20	0	0	0	100
		1,2852	0,9780	1,8421	0,0000
100	5	0	0	30	60
		0,8733	0,9921	0,1638	0,0170
100	10	0	0	10	70
		1,2714	0,9834	0,2189	0,0230
100	20	0	0	0	100
		1,0678	0,8361	0,6627	0,0000

6. Considerações Finais

Os resultados experimentais mostraram que o método heurístico evolutivo HE apresentou desempenho superior em comparação aos métodos reportados na literatura para a solução do problema de programação da produção flow shop com minimização da soma total dos tempos de fluxos das tarefas.

O problema clássico de seqüenciamento de tarefas em um ambiente de produção flow shop tem sido objeto de intensos esforços de pesquisa nos últimos 50 anos e, para fins práticos, tal problema pode ser considerado já resolvido. Entretanto, tendo em vista sua complexidade, ainda permanece como uma direção de pesquisa a busca de métodos heurísticos e metaheurísticos cada vez mais eficazes quanto à qualidade da solução.

A realização da pesquisa relatada neste trabalho foi motivada pelas considerações acima, procurando resgatar as características essenciais de um método metaheuristico, ou seja, adequado equilíbrio entre a qualidade da solução e a eficiência computacional, simplicidade e facilidade de implementação.

7. Referências

Ahmadi, R. H., Bagchi, U. (1990) Improved Lower Bounds for Minimizing the Sum of Completion Times of N Jobs over M Machines, European Journal of Operational Research, Vol. 44 , No. 3, pp. 331-336.

Allahverdi, A., Aldowaisan, T. (2002) New Heuristics to Minimize Total Completion Time in M-Machine Flowshops, International Journal of Production Economics, Vol. 77 , No. 1, pp. 71-83.

Bean, J. C. (1994) Genetic Algorithm and Random Keys for Sequencing and Optimization, ORSA Journal on Computing, Vol. 6, No. 2, pp. 154-160.

Campbell, H. G., Dudek, R. A., Smith, M. L. (1970) A Heuristic Algorithm for N-Job, M-Machine Sequencing Problem, Management Science, Vol. 16, No. 10, pp. 630-637.

Framinan J. M., Leisten, R., Ruiz-Usano R. (2005) Comparison Of Heuristics for Flowtime Minimisation in Permutation Flowshop, Computer and Operations Research, Vol. 32, No.5, pp. 1237-1254.

Framinan, J. M., Leisten, R. (2003) An Efficient Constructive Heuristic for Flowtime Minimisation in Permutation Flow Shop, OMEGA, Vol. 31, No. 4, pp. 311-317.

Garey, M. R., Johnson, D. S., Sethi, R. (1976) The Complexity of Flowshop and Jobshop Scheduling, Mathematics of Operations Research, Vol. 1, No.2, pp.117-129.

Gupta, J. N. D. (1972) Heuristic Algorithms for Multistage Flowshop Scheduling Problem, IIE Transactions, Vol. 4, No. 1, pp.11-18.

Ho, J. C. (1995) Flowshop Sequencing with Mean Flowtime Objective, European Journal of Operational Research, Vol. 81, No. 3, pp. 571-578.

Li, X., Wang, Q., Wu, C. (2009) Efficient Composite Heuristics for Total Flowtime Minimization in Permutation Flow Shops, Omega, Vol. 37, No. 1, pp. 155-164.

Liu, J., Reeves C. R. (2001) Constructive and Composite Heuristic Solutions to the P//∑Ci Scheduling Problem, European Journal of Operational Research, Vol. 132, No. 2, pp. 439-452.

Merkle, D., Middendorf, M. (2000) An Ant Algorithm with a New Pheromone Evaluation Rule for Total Tardiness, In: Proceedings of the Evo Workshops, In: Lecture Notes in Computer Science, Vol. 1803, Springer-Verlag, Berlin, pp.290-299.

Miyazaki, S., Nishiyama, N., Hashimoto, F. (1978) An Adjacent Pairwise Approach to the Mean Flow-Time Scheduling Problem, Journal of the Operations Research Society of Japan, Vol. 21, No. 2, pp. 287-299.

Nagano, M. S., Moccellin, J. V. (2008) Reducing Mean Flow Time in Permutation Flow Shop. Journal of the Operational Research Society, Reducing mean flow time in permutation flow shop, Vol. 59, No. 7, pp. 939-945.

Nawaz, M., Enscore, E. E., Ham, I. (1983) A Heuristic Algorithm for the M-Machine, N-Job Flow-Shop Sequencing Problem, OMEGA, Vol. 11, No.1, pp. 91-95.

Rajendran, C. (1993) Heuristic Algorithm for Scheduling in a Flowshop to Minimise Total Flowtime, International Journal Production Economics, Vol. 29, n. 1, p.65-73.

Rajendran, C., Chaudhuri, D. (1991) An Efficient Heuristic Approach to the Scheduling of Jobs in a Flowshop, European Journal of Operational Research, Vol. 61, No. 1, pp. 318-325.

Rajendran, C., Ziegler H. (2004) Ant-Colony Algorithms for Permutation Flowshop Scheduling to Minimize Makespan/Total Flowtime of Jobs, European Journal of Operational Research, Vol. 155, No. 2, pp. 426-438.

Rajendran, C., Ziegler, H. (1997) An Efficient Heuristic for Scheduling in A Flowshop to Minimize Total Weighted Flowtime Of Jobs, European Journal of Operational Research, Vol. 103, No. 1, pp. 129-138.

Rinnooy Kan, A. H. G. (1976) Machine Scheduling Problems: Classification, Complexity, and Computations. The Hahue, Nijhoff.

Stützle, T. (1998) Applying Iterated Local Search to the Permutation Flowshop Problem. Technical Report, AIDA 98-04, Darmstad University of Technology, Computer Science Department, Intelletics Group, Darmstad, Germany.

Syswerda, G. (1989) Uniform Crossover in Genetic Algorithms. In: International Conference on Genetic Algorithms (ICGA), Virginia, USA, pp. 2–9.

Taillard, E. (1993) Benchmarks for Basic Scheduling Problems. European Journal of Operational Research, Vol. 64, No. 2, pp. 278-285.

Tasgetiren, M. F., Liang, Y. C., Sevkli, M., Gencyilmaz, G. (2007) A Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in the Permutation Flowshop Sequencing Problem, European Journal of Operational Research, Vol. 177, No. 3, pp.1930-1947.

Wang, C., Chu, C., Proth, J. M. (1997) Heuristic Approaches for N/M/F/ΣCi Scheduling Problems, European Journal of Operational Research, Vol. 96, No. 3, pp. 636-644.

Watson, J. P., Barbulescu, L., Whitley L. D., Howe, A. E. (2002) Contrasting Structured and Random Permutation Flowshop Scheduling Problems: Search Space Topology and Algorithm

Performance, Informs Journal on Computing, Vol. 14, No. 2, pp. 98-123.

Woo, D. S., Yim H. S. (1998) A Heuristic Algorithm for Mean Flowtime Objective in Flowshop Scheduling, Computers and Operations Research, Vol. 25, No. 3, pp. 175-182.