Um algoritmo heurístico híbrido para uma classe de problemas de sequenciamento em uma máquina

Mário Henrique de Paiva Perché **Pablo Luiz Araújo Munhoz**
Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto - Departamento de Ciência da Computação
35400-000, Campus Morro do Cruzeiro , Ouro Preto, MG
E-mail: mariohpp@gmail.com, pablo.munhoz@gmail.com, marcone@iceb.ufop.br .

RESUMO

Problemas de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção (PSUMAA) aparecem em muitas aplicações reais, geralmente vinculadas ao uso do sistema de administração *Just in Time* (JIT). Nesse sistema, há a necessidade de organização da produção de forma que tanto antecipações quanto atrasos não sejam recomendados. Uma versão pouco explorada do PSUMAA [1], e objeto deste trabalho, é aquela em que se considera janelas de entrega distintas, tempos de preparação dependentes da sequência de produção e a possibilidade de tempos ociosos entre as execuções de tarefas consecutivas. Sendo este problema da classe NP-difícil [4], propõe-se resolvê-lo por um algoritmo heurístico.

O algoritmo proposto, denominado GIVPR, é baseado em GRASP, *Iterated Local Search* - ILS, *Variable Neighborhood Descent* - VND [3] e Reconexão por Caminhos - RC [2]. Na primeira fase deste algoritmo gera-se uma solução inicial parcialmente gulosa, seguindo as ideias da fase de construção GRASP. A segunda fase consiste em refinar essa solução usando-se a metaheurística ILS tendo como busca local desta o procedimento VND. A exploração do espaço de busca é feita por movimentos baseados em trocas de tarefas e realocação de uma tarefa ou de um bloco de tarefas para outra posição da sequência. As perturbações do ILS são definidas como trocas aleatórias entre duas tarefas quaisquer. Caso o nível de perturbação seja *Npert*, são realizadas $Npert + 1$ trocas. Esse nível é iniciado em 1 e é aumentado em uma unidade sempre que não haja melhora na melhor solução após a análise de pelo menos 10% da vizinhança. Se uma solução melhor é gerada, o nível de pertubação é reiniciado. O algoritmo termina quando é atingido um número máximo de iterações sem melhora.

Durante a segunda fase, um conjunto de soluções elite é formado. Para integrar esse conjunto uma solução deve atender a um dos seguintes critérios: 1) ser melhor que a melhor solução do conjunto; 2) ser melhor que a pior solução do conjunto e diferenciada das demais de um determinado percentual de atributos. O atributo considerado é a posição da tarefa em ambas as soluções. Com isso, nesse grupo há soluções de boa qualidade e diversificadas entre si.

A estratégia RC é acionada periodicamente durante o ILS. São sorteadas duas soluções do conjunto elite, sendo uma delas definida como Solução Guia e a outra como Solução Corrente. Atributos da Solução Guia são adicionados um a um na Solução Corrente até que a Solução Corrente se torne a Guia. Os atributos utilizados são arcos formados por duas tarefas adjacentes. A RC é executada tanto da pior solução selecionada para a melhor (*Forward Path Relinking*), quanto da melhor para a pior (*Backward Path Relinking*). Durante essa trajetória a expectativa é que melhores soluções sejam geradas. Além dessa utilização, elementos do conjunto elite são submetidos periodicamente a um esforço de busca utilizando-se os operadores *crossover* PMX, OX e CX. Os filhos gerados pela aplicação de um desses operadores são submetidos a uma busca local baseada em VND randômico com o objetivo de melhorar a qualidade das soluções.

Para testar o algoritmo proposto foram utilizados os problemas-testes de [1], envolvendo 8, 9, 10, 11, 12, 15, 20, 25, 30 e 40 tarefas. A implementação foi feita utilizando a linguagem C++

e o ambiente de desenvolvimento Borland C++ Builder 5. O computador utilizado para os testes foi um Core 2 Quad com clock de 2,4 GHz, 4 GB de memória RAM e sistema operacional Windows Vista Ultimate. Apesar de o processador possuir quatro núcleos, esse recurso de multiprocessamento não foi utilizado para a execução dos testes.

Na Tabela 1 são apresentados os resultados encontrados por GIVPR, comparando-os com um algoritmo da literatura [1], bem como com aqueles gerados pelo otimizador CPLEX, versão 11, aplicado a um modelo de programação matemática da literatura.

São aplicadas três medidas de desempenho, dadas pelas equações: $imp_i^{best} = \frac{f_i^{GJr^*} - f_i^{GIVPR^*}}{f_i^{GJr^*}}$, $imp_i^{avg} = \frac{\bar{f}_i^{GJr} - \bar{f}_i^{GTSPR}}{\bar{f}_i^{GJr}}$ e $gap_i^{avg} = \frac{\bar{f}_i^{GIVPR} - f_i^*}{f_i^*}$, que verifica o quanto o algoritmo proposto superou o melhor resultado de [1], quantifica de quanto foi o desvio médio das soluções geradas pelo método GIVPR em relação à melhor solução encontrada por [1] e calcula o desvio das soluções médias do algoritmo em relação às melhores soluções existentes, respectivamente. Nestas equações, para cada problema-teste i do grupo, $f_i^{GIVPR^*}$ representa o melhor valor encontrado pelo algoritmo proposto, $f_i^{GJr^*}$ é o melhor valor encontrado por [1], f_i^* é o valor da melhor solução conhecida, $\bar{f}_i^{GJr}$ representa o valor médio obtido pelo algoritmo de [1] e $\bar{f}_i^{GIVPR}$ é o valor médio encontrado pelo algoritmo proposto.

Tabela 1: Comparação entre CPLEX, Gomes Jr. e GIVPR

Tar.	imp_i^{best} (%)	imp_i^{avg} (%)	gap_i^{avg} (%)	CPLEX (s)	Gomes Jr. (s)	GIVPR (s)
8	0,00	0,00	0,03	3,04	0,04	0,18
9	0,00	0,04	0,02	78,23	0,07	0,33
10	0,00	0,01	0,01	107,39	0,11	0,51
11	0,00	0,09	0,03	1615,07	0,20	0,98
12	0,00	0,14	0,07	1819,46	0,29	1,58
15	0,00	0,72	0,75	-	0,94	5,25
20	0,00	1,18	0,47	-	4,35	28,39
25	0,00	1,56	0,75	-	13,29	88,58
30	0,00	2,72	0,63	-	40,07	340,36
40	0,03	3,43	1,24	-	155,79	1173,30

Como se observa, o algoritmo GIVPR consegue encontrar todas as melhores soluções de [1] e melhorar em 0,03% os resultados deste em problemas-teste envolvendo 40 tarefas. Além disso, produz resultados médios melhores que o de [1], em até 3,43%, e é robusto por gerar soluções finais de variabilidade média inferior a 1,24%. Por outro lado, há um aumento do tempo de processamento demandado por GIVPR em relação ao de [1], porém este tempo é muito menor que aquele gasto pelo CPLEX e inferior ao horizonte de planejamento (tipicamente uma semana).

Palavras-chave: *Sequenciamento em uma máquina, Metaheurísticas, GRASP, Iterated Local Search, Descida em vizinhança variável, Reconexão por Caminhos*

Referências

[1] A.C. Gomes Júnior et al. Um método heurístico híbrido para a resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional*, p. 1649-1660, 2007.

[2] F. Glover Tabu search and adaptive memory programming - advances, applications and challenges. *Interfaces in computer science and operations research*, 1996. p. 1-75.

[3] F. Glover; G.A. Kochenberger, "Handbook of Metaheuritics", Kluwer, 2003.

[4] G. Wan; B.P.C. Yen, Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, v. 142, pp. 271-281, 2002.