Anais do IX Congresso Brasileiro de Redes Neurais / Inteligência Computacional (IX CBRN)
Ouro Preto 25-28 de Outubro de 2009

UM NOVO ALGORITMO HEURÍSTICO BASEADO EM ILS PARA UMA CLASSE DE PROBLEMAS DE SEQUENCIAMENTO

Pablo Luiz Araújo Munhoz*, Mário Henrique de Paiva Perché*, Marcone Jamilson Freitas Souza*

* *Universidade Federal de Ouro Preto*
Departamento de Computação
Ouro Preto, Minas Gerais, Brasil

Emails: pablomunhoz@iceb.ufop.br, mariohpp@iceb.ufop.br, marcone@iceb.ufop.br

Abstract— This paper is focused in a class of problems of single-machine scheduling with earliness and tardiness penalties. Due windows and sequence dependent setup are considered too. To solve the problem, it is proposed an algorithm based on Iterated Local Search (ILS) metaheuristic with Path Relinking. To seek for better solutions are used moves based on relocation and swap of jobs, as well relocation of block of jobs. ILS uses Variable Neighborhood Descent as a method of local search. During the search realized by ILS, a set of elite solutions is stored and the Path Relinking is applied on this set. In addition, operators of search based on crossover are applied to pairs of solutions of this set and their descendants are submitted by a random descent method. The performance of the proposed algorithm is compared with an another heuristic algorithm found in the literature.

Keywords— Single Machine Scheduling, Iterated Local Search.

Resumo— Este trabalho tem seu foco em uma classe de problemas de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. No problema abordado são considerados tempos de preparação da máquina dependentes da sequência de produção, bem como a existência de janelas de entrega distintas. Para sua resolução, propõe-se um algoritmo baseado na metaheurística *Iterated Local Search* - ILS, combinado com Reconexão por Caminhos. A busca local é feita pelo método *Variable Neighborhood Descent* e durante a mesma é armazenado um conjunto de soluções elite, sendo sobre essas aplicada a Reconexão por Caminhos. Adicionalmente, operadores de busca baseados em *crossover* são aplicados a pares de soluções desse conjunto e seus descendentes são submetidos a uma busca local pelo Método da Descida Randômica. O desempenho do algoritmo proposto é comparado com um algoritmo heurístico encontrado na literatura.

Palavras-chave— Sequenciamento em uma máquina; Iterated Local Search; Planejamento da produção.

1 Introdução

O problema de sequenciamento em uma máquina com minimização das penalidades por antecipação e atraso da produção (*Single Machine Scheduling for Minimizing Earliness and Tardiness Penalties*), denotado por PSUMAA, modela um dos mais simples problemas de planejamento da produção no contexto da filosofia *just-in-time*.

Além da importância prática desse problema, com inúmeras aplicações nas indústrias (Allahverdi et al., 1999), trata-se de um problema difícil de ser resolvido na otimalidade em tempos computacionais aceitáveis, visto pertencer à classe NP-difícil (Wan and Yen, 2002). Essa conjunção de aplicabilidade e dificuldade de resolução na otimalidade tem motivado os pesquisadores a desenvolverem algoritmos eficientes para resolvê-lo.

No presente trabalho, propõe-se um novo algoritmo, denominado GIVPR, que combina os procedimentos heurísticos GRASP (Feo and Resende, 1995), ILS (Lourenço et al., 2003), VND (Mladenović and Hansen, 1997) e a técnica Reconexão por Caminhos (Glover, 1996). Além disso, operadores *crossover* são usados como um esforço adicional de busca. Este algoritmo é uma extensão daquele de Gomes Júnior et al. (2007), por incluir os módulos adicionais de Reconexão por Caminhos e aplicação de operadores *crossover*. O algoritmo proposto é comparado com o de Souza et al. (2008), o qual, por sua vez, se mostrou superior ao de Gomes Júnior et al. (2007).

Este trabalho está organizado como segue. Na Seção 2 é apresentada a metodologia heurística proposta. Na Seção 3 são apresentados e analisados os resultados obtidos, enquanto a última Seção conclui o trabalho.

2 Solução Implementada

O algoritmo proposto, denominado GIVPR, baseia-se em metaheurísticas, as quais, ao contrário dos procedimentos heurísticos clássicos, têm mecanismos que possibilitam escapar dos ótimos locais. O algoritmo GIVPR combina os procedimentos GRASP, ILS, VND, Reconexão por Caminhos e operadores *crossover*. As subseções seguintes detalham seu funcionamento.

2.1 Representação de uma solução

Uma solução do PSUMAA é representada por um vetor v de n elementos, onde cada posição $i = 1, 2, ..., n$ indica a ordem de produção da tarefa v_i da sequência. Assim, na sequência $v = (5, 3, 2, 1, 4, 6)$, a tarefa 5 é a primeira a ser realizada e a 6, a última.

2.2 Estrutura de Vizinhança

São usados três tipos de movimento para explorar o espaço de soluções: troca da ordem de processamento de duas tarefas da sequência de produção, realocação de uma tarefa para outra posição na sequência de produção e realocação de um bloco de k tarefas. Esses movimentos definem, respectivamente, as estruturas de vizinhança N^T, N^R e N^{Or}.

2.3 Função de Avaliação

Uma solução v é avaliada pela função f a seguir, a qual deve ser minimizada:

$$f(v) = \sum_{i=1}^{n} (\alpha_i e_i + \beta_i t_i) \quad (1)$$

em que e_i e t_i indicam, respectivamente, o tempo de antecipação e atraso da tarefa i e α_i e β_i são as penalidades respectivas. Para determinar as datas ótimas de início de processamento das tarefas da sequência dada e, consequentemente, determinar os valores e_i e t_i acima, utiliza-se o algoritmo ADDOIP de Gomes Júnior et al. (2007).

2.4 Algoritmo GIVPR

O pseudocódigo do algoritmo proposto, denominado GIVPR, é apresentado no Algoritmo 1.

Algoritmo 1: GIVPR(v)

```
v0 ← Construção_GRASP();
v ← VND(v0);
enquanto (Critério de parada não satisfeito)
faça
    v' ← Perturbação(v, nível);
    v'' ← VND(v');
    v ← CritérioAceitação(v, v'', nível);
    se (v satisfaz as condições de inserção no GE)
    então
        GE ← GE ⋃ v;
    fim
    se
    (Condição de Ativação Crossover satisfeita)
    então
        v' ← Elemento Aleatório do GE;
        v'' ← Elemento Aleatório do GE;
        v ← crossover( v', v'' );
    fim
    se (Condição de Ativação PR satisfeita)
    então
        v' ← Elemento Aleatório do GE;
        v'' ← Elemento Aleatório do GE;
        v ← PR( v', v'' );
    fim
fim
return (v)
```

O algoritmo GIVPR utiliza a fase de construção GRASP para gerar uma solução inicial, sendo este procedimento descrito na Seção 2.5. O método de busca local do algoritmo proposto é o *Variable Neighborhood Descent* - VND, o qual é detalhado na Seção 2.6. A adaptação do ILS é apresentada na Seção 2.7. Durante a exploração do espaço de busca, é armazenado um conjunto de soluções de boa qualidade e diversificadas entre si, denominado grupo elite (GE). O algoritmo GIVPR também aciona a Reconexão por Caminhos, descrita na Seção 2.8, como mecanismo de intensificação, bem como ativa a utilização de operadores *crossover* periodicamente para combinar soluções provenientes do grupo elite. A forma de funcionamento desses operadores *crossover* está descrita na Seção 2.9.

2.5 Construção da Solução Inicial

Uma solução inicial s_0 parcialmente gulosa é construída elemento por elemento, utilizando sucessivamente a fase de construção do procedimento GRASP (Feo and Resende, 1995). Como função guia para a construção da solução utiliza-se a heurística *Earliest Due Date* - EDD (Baker and Scudder, 1990), que sequencia as operações em relação às datas de início da janela de entrega desejadas em ordem crescente.

Para a construção da solução inicial uma lista de

candidatos (LC) com todas as tarefas ainda não sequenciadas é criada. Estas são ordenadas pela data de início da janela de entrega, segundo o procedimento heurístico EDD.

Uma lista restrita de candidatos (LRC) é construída a partir da LC, contendo as tarefas melhores classificadas em relação à data de início da janela de entrega. O tamanho dessa lista é definido pelo parâmetro escolhido em um conjunto tal como em Gomes Júnior et al. (2007). A seguir é escolhida aleatoriamente uma tarefa desta lista, sendo então adicionada à solução parcial. A fase de construção termina quando todas as tarefas forem alocadas.

2.6 Busca Local

A busca local do algoritmo GIVPR é feita pelo procedimento heurístico *Variable Neighborhood Descent* - VND (Mladenović and Hansen, 1997). Esse procedimento trabalha com as estruturas de vizinhança N^R, N^T e N^{Or}, nesta ordem. Inicialmente gera-se um vizinho $v' \epsilon N^R$ da solução corrente v. Caso essa solução vizinha v' seja melhor que a solução corrente v, a solução corrente é atualizada para a solução vizinha. Continua-se aplicando a primeira estrutura de vizinhança até que o número máximo *iterVND* de iterações sem melhora seja alcançado. Quando isto ocorre, altera-se a estrutura de vizinhança para N^T e então, a mesma ideia da primeira estrutura de vizinhança é aplicada. Se houver melhora nesta nova estrutura, retorna-se à primeira vizinhança e reinicia-se o processo; caso contrário, passa-se para a vizinhança seguinte N^{Or}. O procedimento termina quando não há melhora nas três vizinhanças. Destaca-se que o procedimento não necessariamente retorna um ótimo local com relação às três vizinhanças, uma vez que nem todos os vizinhos são analisados.

2.7 Algoritmo GVIPR

Para aplicar um algoritmo ILS, quatro componentes têm que ser especificadas: I) um procedimento que gera uma solução inicial s_0 para o problema; II) um procedimento de busca local com objetivo de retornar uma solução melhorada *s"*; III) um procedimento de perturbação, que modifica a solução corrente s guiando-a a uma solução intermediária *s'* e IV) um procedimento que avalia a solução *s"*, obtida com a execução do procedimento de busca local, em relação a melhor solução encontrada até então.

No algoritmo GIVPR, o VND é utilizado como método de busca local. Além disso, a Reconexão por Caminhos (*Path Relinking* - PR), é usada como método de intensificação. Adota-se *iterMax* $(1,4 \times n)$ iterações sem melhora como critério de parada do algoritmo. Move-se de uma solução para outra somente se o ótimo local encontrado for melhor que o ótimo local corrente, ou seja, uma solução v é aceita como nova solução corrente,somente se *f(v") < f(v)*.

As perturbações no GIVPR são feitas por meio de trocas aleatórias entre duas tarefas quaisquer da solução. Caso o nível de perturbação seja igual a *nPerturb*, então são feitas *nPerturb* + 1 trocas aleatórias.

O nível de perturbação é aumentado em uma unidade quando a busca local não resultar em melhoria da solução corrente e quando no mínimo 10% da vizinhança dessa solução tenha sido explorada. Quando há melhora, o nível de perturbação volta para seu nível mínimo, isto é, 1, voltando a se repetir o processo.

Durante a execução do algoritmo, um grupo de soluções elite (GE) é criado com o intuito de armazenar soluções de boa qualidade e diversificadas entre si. Para fazer parte do GE, cada solução deve satisfazer a pelo menos um dos dois seguintes critérios: 1) Ser melhor que a melhor solução do grupo elite; 2) ser melhor que a pior solução e ter, no mínimo, um percentual de diferença de atributos das demais soluções do grupo elite, dado por *divElite*. O atributo de comparação considerado é a posição de uma tarefa na solução. Após o grupo elite formado, ao adicionar uma solução, a de pior valor é removida.

A cada *iterPR* $(0,75 \times iterMax)$, a estratégia PR é ativada. Duas soluções são escolhidas aleatoriamente dentre o conjunto GE e a Reconexão é aplicada de forma bidirecional.

Como um esforço adicional de busca, também são aplicados operadores de busca crossover. Esses operadores são aplicados a cada *iterCross* $(0,20 \times IterMax)$ iterações sem melhora. São selecionadas aleatoriamente duas soluções do grupo elite GE e gerados descendentes dessas a partir da utilização de um dos operadores, escolhidos de forma aleatória.

2.8 Reconexão por Caminhos

Reconexão por Caminhos é uma estratégia de intensificação normalmente aplicada ou como pós-otimização, ou como refinamento de ótimos locais obtidos durante a busca. Dado um par de soluções, o objetivo da técnica é a partir de uma delas, denominada solução base, chegar à outra, denominada solução guia, por meio da adição na solução base de atributos da solução guia. Para cada par de soluções, caminha-se de forma bidirecional,

isto é, tanto da pior solução para a melhor (dita Reconexão por Caminhos Progressiva - *Forward Path Relinking*), quanto da melhor para a pior solução (dita Reconexão por Caminhos Regressiva - *Backward Path Relinking*). Determinadas as soluções base e guia, o procedimento é executado inserindo gradativamente um atributo da solução guia na solução base. Cada tarefa i da solução guia é inserida na solução base, na ordem e na posição correspondente da solução guia. Após cada inserção, a tarefa i que já se encontrava na solução base é removida, gerando uma nova sequência para cada atributo específico da solução guia.

Ao final desses passos, o atributo inserido que produziu a melhor solução é incorporado em definitivo à solução base. A seguir são novamente adicionados à solução base os atributos da solução guia ainda não inseridos e o procedimento prossegue como anteriormente, encerrando-se quando se alcança a solução guia, isto é, quando a solução base passa a ter todos os atributos da solução guia.

2.9 Operadores crossover

Nos Algoritmos Genéticos (AGs), cada cromossomo (indivíduo da população) está associado a uma solução do problema e cada gene está associado a uma componente da solução. No caso do PSUMAA, cada gene representa uma tarefa da sequência de produção.

Para explorar o espaço de soluções, os AGs usam os operadores de busca crossover e mutação. Neste trabalho, foram utilizados os operadores crossover PMX, OX e CX, para combinar soluções provenientes do grupo elite formadas durante a busca, sendo ativados após $0,20 \times IterMax$ iterações sem melhora do algoritmo. A cada descendente gerado, aplica-se o Método Randômico de Descida para refiná-los, usando movimentos de realocação.

3 Resultados Computacionais e Análise

Para testar o algoritmo proposto foram utilizados os problemas-testes de Gomes Júnior et al. (2007) envolvendo 8, 9, 10, 11, 12, 15, 20, 25, 30 e 40 tarefas. Para cada número de tarefas existem 12 problemas-teste, totalizando 120 problemas. 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, com 4 GB de memória RAM e utilizando o sistema operacional Windows Vista Ultimate. Apesar de o processador possuir quatro núcleos, esse recurso de multiprocessamento não foi utilizado para a execução dos testes.

São aplicadas duas medidas de desempenho, dadas pelas equações (2) e (3). Nestas equações, para cada problema-teste i do grupo, $f_i^{GIVPR^*}$ representa o melhor valor encontrado pelo algoritmo GIVPR proposto, $f_i^{Souza^*}$ é o melhor valor encontrado por Souza et al. (2008), f_i^* é o valor da melhor solução conhecida, $\bar{f}_i^{Souza}$ representa o valor médio obtido pelo algoritmo de Souza et al. (2008) em trinta execuções e $\bar{f}_i^{GIVPR}$ é o valor médio encontrado pelo algoritmo proposto nas diversas execuções para o i-ésimo problema-teste.

$$imp_i^{best} = \frac{f_i^{GIVPR^*} - f_i^{Souza^*}}{f_i^{Souza^*}} \quad (2)$$

$$imp_i^{avg} = \frac{\bar{f}_i^{GIVPR} - \bar{f}_i^{Souza}}{\bar{f}_i^{Souza}} \quad (3)$$

Na primeira medida de desempenho, verifica-se o quanto o algoritmo proposto superou o melhor resultado de Souza et al. (2008). Para tanto, para cada problema-teste, aplica-se a equação (2), sendo apresentados a média percentual desses valores para cada grupo de problemas-teste com mesmo número de tarefas.

Na segunda medida de desempenho, quantifica-se de quanto o algoritmo GVIPR melhorou os resultados médios do algoritmo de Souza et al. (2008). Nesta medida, comparam-se os algoritmos quanto à variabilidade das soluções finais.

A Tabela 1 resume os resultados encontrados. Nesta tabela, a primeira coluna mostra o número de tarefas envolvidas no problema-teste, enquanto a segunda e terceira colunas mostram os resultados da aplicação das medidas de desempenho anteriores. As duas últimas colunas mostram os tempos computacionais despendidos, em segundos.

Tabela 1: Comparação Souza et al. × GIVPR

Tar.	imp_i^{best}	imp_i^{avg}	Souza (s)	GIVPR (s)
8	0,00	-0,03	0,02	0,18
9	0,00	-0,02	0,04	0,33
10	0,00	-0,01	0,06	0,51
11	0,00	-0,02	0,10	0,98
12	0,00	0,02	0,15	1,58
15	0,00	0,47	0,46	5,25
20	0,00	0,63	2,05	28,39
25	0,00	0,90	6,62	88,58
30	0,00	1,87	18,66	340,36
40	0,11	2,49	84,16	1173,30

Pela Tabela 1, observa-se que em todos os casos a melhor solução foi encontrada, e que em problemas-teste de 40 tarefas GIVPR conseguiu obter uma melhora de 0,11% em relação ao resultado de Souza et al. (2008). Em relação aos resultados médios,

verifica-se que com até 11 tarefas, o algoritmo de Souza et al. (2008) mostrou-se ligeiramente superior; porém, nos demais problemas-teste, o algoritmo GIVPR consegue superá-lo, conseguindo uma diminuição na variabilidade das soluções em até 2,49%.

4 Conclusões

Este trabalho tratou o problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção (PSUMAA), considerando janelas de entrega e tempo de preparação da máquina dependente da sequência de produção.

Para resolvê-lo foi desenvolvido um algoritmo heurístico baseado em *Iterated Local Search* e Reconexão por Caminhos, denominado GIVPR. Esse algoritmo usa a fase de construção GRASP para gerar uma solução inicial; o *Variable Neiborhood Descent* como método de busca local do ILS e a Reconexão por Caminhos como técnica de intensificação aplicada periodicamente durante a exploração do espaço de busca. Adicionalmente, GVIPR aplica periodicamente operadores genéticos de busca a um conjunto de soluções elite objetivando gerar soluções de qualidade para o problema.

Os testes com o algoritmo GIVPR foram feitos utilizando problemas-teste envolvendo até 40 tarefas. O algoritmo proposto conseguiu encontrar todos os resultados da literatura e ainda conseguiu superar os melhores resultados existentes para 40 tarefas. Além disso, em relação às soluções médias, gerou resultados com menor variabilidade. Apesar de o tempo computacional ser maior que o despendido pelo algoritmo de Souza et al. (2008), ainda é competitivo, tendo em vista que o horizonte de planejamento de uma indústria produtiva é de cerca de uma semana.

Como trabalho futuro aponta-se a paralelização do algoritmo em vários núcleos e o estudo de propriedades que possam ser aplicadas ao problema, de forma a reduzir o espaço de busca e, conseqüentemente, o tempo de processamento do algoritmo.

Agradecimentos

Os autores agradecem ao CNPq e à FAPEMIG pelo apoio dado à execução do presente trabalho.

Referências

Allahverdi, A., Gupta, J. N. and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations, *Omega, International Journal of Management Science* **27**: 219–239.

Baker, K. R. and Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: A review, *Operations Research* **38**: 22–36.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization* **6**: 109–133.

Glover, F. (1996). Tabu search and adaptive memory programming - advances, applications and challenges, *in* R. Barr, R. Helgason and J. Kennington (eds), *Interfaces in Computer Sciences and Operations Research*, Kluwer Academic Publishers, pp. 1–75.

Gomes Júnior, A. C., Carvalho, C. R. V., Munhoz, P. L. A. and Souza, M. J. F. (2007). 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* **1**: 1649–1660.

Lourenço, H. R., Martin, O. C. and Stützle, T. (2003). Iterated local search, *in* F. Glover and G. A. Kochenberger (eds), *Handbook of Metaheuristics*, Kluwer Academic Publishers, chapter 11, pp. 321–353.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search, *Computers and Operations Research* **24**(11): 1097–1100.

Souza, M. J. F., Penna, P. H. V. and Gonçalves, F. A. C. A. (2008). Grasp, vnd, busca tabu e reconexão por caminhos para o problema de seqüenciamento em uma máquina com tempos de preparação dependentes da seqüência da produção, janelas de entrega distintas e penalidades por antecipação e atraso da produção, *Anais do XL Simpósio Brasileiro de Pesquisa Operacional* **1**: 1320–1331.

Wan, G. and Yen, B. P.-C. (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties, *European Journal of Operational Research* **142**: 271–281.