UMA HEURÍSTICA HÍBRIDA PARA O PROBLEMA DE SEQUENCIAMENTO DE TAREFAS EM MÁQUINAS PARALELAS NÃO RELACIONADAS

João Paulo de Castro Martins Nogueira, José Elias Claudio Arroyo, Luciana Brugiolo Gonçalves
Departamento de Informática - Universidade Federal de Viçosa
Centro de Ciências Exatas e Tecnológicas, Campus da UFV, 36570-000, Viçosa - MG
jpcmnogueira@gmail.com, jarroyo@dpi.ufv.br, lbrugiolo@ufv.br

RESUMO

Este trabalho trata do Problema de Sequenciamento de Tarefas num ambiente com Maquinas Paralelas não Relacionadas onde o objetivo é minimizar a soma das penalizações por antecipação e atraso. São considerados tempos de *setup* dependentes da sequência das tarefas, bem como a possibilidade de inserção de tempos de ociosidade. Para resolver o problema é apresentada uma heurística híbrida baseadas na metaheurística GRASP e *Path Relinking*. Para cada sequência de tarefas obtidas pela heurística é utilizado um algoritmo de tempo polinomial para determinar a data ótima de início das tarefas. A partir dos resultados foi possível observar a boa qualidade das soluções obtidas pela heurística, destacando a contribuição do procedimento de *Path Relinking*. Os resultados são analisados e confirmados por testes estatísticos.
PALAVRAS CHAVE. Escalonamento, GRASP, *Path Relinking*.

Áreas Principais: Metaheurísticas, Otimização Combinatória.

ABSTRACT

This paper deals with unrelated parallel machine scheduling problem where the objective is to reduce the total weighted earliness and tardiness penalties. Sequence dependent setup time and idle time are considered. To solve this problem, a hibrid heuristic is presented. This approach is based in the metaheuristic GRASP and Path Relinking. For each job sequence generated by the heuristic, a polinomial time algorithm is used to determine the optimal date for each job. The computational experiments carried out show that the heuristic was able to find good solutions. Besides that, it was possible to identify the contribution of the Path Relinking precedure. Statistical tests confirm the reported results.
KEY WORDS: Scheduling, GRASP, Path Relinking.

Main areas: Metaheuristics, Combinatorial Optimization.

1 Introdução

Este trabalho trata do Problema de Sequenciamento de Tarefas num ambiente com Máquinas Paralelas não Relacionadas - MPNR. Na literatura, este problema tem sido resolvido considerando diferentes funções de otimização, tais como: minimização do *makespam* (Guo *et al*., 2007; Vallada e Ruiz, 2011), do tempo médio de fluxo (Fowler *et al*., 2003), ou ainda, da soma dos atrasos (Tamimi e Rajan, 1997). Neste trabalho é considerado como objetivo a minimização das penalidades por antecipação e atraso, onde penaliza-se a conclusão das tarefas antes do instante em que essas são requeridas, além de penalidade associada ao atraso. Este objetivo se justifica pela adoção por muitas empresas da filosofia *Just-In-Time* (JIT). Esta filosofia emprega que toda produção seja concluída o mais próximo possível de sua data de entrega evitando, desta forma, encargos financeiros extras. Enquanto o atraso pode gerar multas contratuais, a antecipação pode

causar custos associados à necessidade adiantada de capital, espaço para armazenamento e/ou recursos para manter ou gerenciar os estoques.

No problema abordado, um conjunto $N = \{1, 2, \ldots, n\}$ das tarefas que devem ser processadas em um conjunto de máquinas $M = \{1, 2, \ldots, m\}$. Cada tarefa deve ser executada em uma das máquinas e para cada máquina deve-se determinar a ordem de execução (sequência) das tarefas associadas a ela. Como trata-se de um cenário envolvendo um conjunto de máquinas não relacionadas, o tempo de processamento de uma tarefa depende da máquina a qual esta tarefa foi atribuída. Este tempo é definido *a priori* e pode ser denotado por p_{ik}, tempo de processamento de uma tarefa i se a mesma for executada pela máquina k. Todas as tarefas estão disponíveis no instante inicial do sequenciamento e, para cada tarefa i, d_i representa a sua data de entrega.

Neste contexto, considerando penalização por antecipação e atraso onde cada tarefa possui uma data prevista para sua entrega, uma estratégia que pode propiciar a obtenção de soluções de boa qualidade é a inserção de períodos de ociosidade (*idle time*) entre tarefas consecutivas a serem executadas por uma dada máquina. Em algumas situações, pode ser vantajoso que por algum período de tempo a máquina fique sem executar nenhuma tarefa. Segundo França Filho (2007), em determinadas ocasiões a inserção de ociosidade pode gerar custos maiores do que a própria antecipação na conclusão das tarefas. Entretanto, em outros casos, mesmo com tarefas disponíveis para processamento, vale a pena manter a máquina ociosa.

Uma outra característica tratada neste trabalho é o tempo de *setup*, um período de tempo não produtivo inserido após a execução de uma tarefa e que é necessário para a preparação das máquinas de modo a deixá-las prontas para as próximas tarefas na sequência. Então, o tempo de *setup* necessário na máquina k para o processamento da tarefa j depois da execução tarefa i será denotado por s_{ijk}.

Uma revisão bibliográfica sobre o problema de sequenciamento de tarefas em MPNR foi apresentado por Cheng e Sin (1990). Em Armacost e Salem (1999) e Guo *et al*. (2007) os autores estudaram o problema em relação a minimização do *makespam*, o tempo máximo necessário para a conclusão das tarefas (C_{max}). Vallada e Ruiz (2011) abordam a minimização do *makespam* e consideram *setup time* dependente da sequencia de tarefas. Já Logendran e Saputra (1999) abordam o problema utilizando *setup time* dependentes da sequencia, onde o objetivo é minimizar as penalidades de atraso em relação as datas de entrega. Liaee e Emmons (1997) relatam que o problema de sequenciamento de tarefas em uma máquina com tempo de *setup* dependentes da sequência é classificado como um problema NP-difícil, esta afirmação é apoiada pelo review de Potts e Kovalyov (2000).

Neste trabalho, para tratar o problema de sequenciamento de tarefas em MPNR é proposto um algoritmo baseado na metaheurística GRASP proposta por Feo e Resende (1995) (*Greedy randomized adaptive search procedure*), utilizando a estratégia de *Path Relinking* (PR) de Glover (1996) como um procedimento de intensificação. A heurística GRASP é um método estocástico que tem mostrado bom desempenho na resolução de problemas de otimização combinatória, como mostra Festa e Resende (2009). O objetivo deste trabalho é avaliar a eficiência do PR quando utilizado juntamente com a heurística GRASP para o Problema de Sequenciamento de Tarefas num ambiente com Máquinas Paralelas não Relacionadas.

Este artigo está organizado da seguinte forma. Na Seção 2 é apresentada uma formulação matemática para o problema. Na Seção 3 é descrito o algoritmo proposto e cada um de seus componentes. Os resultados são apresentados na Seção 4. As conclusões e considerações finais estão na Seção 5.

2 Formulação Matemática

Considere o conjunto de tarefas $N = \{1, 2, \ldots, n\}$, onde cada tarefa deve ser processada por exatamente uma das máquinas do conjunto $M = \{1, 2, \ldots, m\}$. Todas as tarefas estão disponíveis para processamento no instante inicial (tempo zero). Para cada tarefa $i \in N$ são

fornecidos os tempos de processamento nas máquinas $k \in M$, denotados por p_{ik}. Cada tarefa i possui também uma data de entrega d_i, uma penalidade de atraso α_i e uma penalidade de adiantamento β_i. Considerando C_i o instante de conclusão da tarefa i, se $C_i < d_i$ então a tarefa i está adiantada (com relação a sua data de entrega). Este adiantamento é calculado por $E_i = d_i - C_i$ e a penalidade associada a este adiantamento é definida como $\beta_i \times E_i$. Porém, se $C_i > d_i$, a tarefa i está atrasada em $T_i = C_i - d_i$ unidades de tempo e a penalidade de atraso é igual a $\alpha_i \times T_i$. Para evitar penalidades, C_i deveria ser igual a d_i.

Neste trabalho são considerados tempos de preparação (tempo de *setup*) dependentes da sequência. Finalizado o processamento de uma tarefa i na máquina k, antes de iniciar o processamento da tarefa j, exite um tempo de *setup* s_{ijk}. Não é considerado tempo de *setup* quando a máquina executa a primeira tarefa da sequência. Uma máquina só e capaz de processar uma tarefa de cada vez e não é permitida a interrupções durante o processamento das tarefas. É permitida a inclusão de tempos de ociosidade (*idle time*) nas máquinas, ou seja, em alguns períodos de tempo as máquinas podem ficar sem executar nenhuma tarefa.

A formulação matemática apresentada nesta seção é baseada no modelo de programação linear inteira mista (PLIM) apresentado por Arenales *et al*. (2007) para o problema de programação da produção em máquinas paralelas. A função objetivo foi adaptada para considerar uma penalização por cada unidade de tempo de adiantamento ou atraso.

Para esta formulação, considere a variável binária x_{ijk} que assume valor 1 se a tarefa i precede a tarefa j na máquina k, 0 caso contrário. A variável C_{ik} registra o instante de conclusão do processamento da tarefa i na máquina k. Além destas variáveis, E_i e T_i representam, respectivamente, o adiantamento e o atraso associado a uma tarefa i, onde $E_i = max(d_i - C_i, 0)$ e $T_i = max(C_i - d_i, 0)$. Para marcar o início e o fim de uma sequência de processamento em cada máquina foi definida a tarefa fictícia 0. O modelo de programação linear é apresentado a seguir.

$$\min \sum_{i=1}^{n} (\alpha_i T_i + \beta_i E_i), \tag{1}$$

Sujeito a:

$$\sum_{k=1}^{m} \sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ijk} = 1, \qquad j = 1, ..., n \tag{2}$$

$$\sum_{j=1}^{n} x_{0jk} \leq 1, \qquad k = 1, ..., m \tag{3}$$

$$\sum_{\substack{i=0 \\ i \neq h}}^{n} x_{ihk} - \sum_{\substack{j=0 \\ j \neq h}}^{n} x_{hjk} = 0, \qquad h = 1, ..., n \text{ e } k = 1, ..., m \tag{4}$$

$$C_{0k} = 0 \qquad k = 1, \cdots, m \tag{5}$$

$$C_{jk} \geq C_{ik} - M + (p_{jk} + s_{ijk} + M) x_{ijk}, \qquad i = 0, ..., n; j = 1, ..., n \text{ e } k = 1, ..., m \tag{6}$$

$$E_i \geq d_i - C_{ik} \qquad i = 1, ..., n \text{ e } k = 1, ..., m \tag{7}$$

$$T_i \geq C_{ik} - d_i \qquad i = 1, ..., n \text{ e } k = 1, ..., m \tag{8}$$

$$T_i \geq 0, E_i \geq 0 \qquad i = 1, ..., n \tag{9}$$

$$x_{ijk} \in \{0, 1\} \qquad i, j = 0, ..., n \text{ e } k = 1, ..., m \tag{10}$$

A função objetivo (1) minimiza a soma das penalidades por atraso ou adiantamento das tarefas. As restrições (2,3,4) asseguram uma sequência exclusiva de tarefas para cada uma das máquinas. Sendo que as restrições (2) indicam que cada tarefa j na máquina k tenham apenas uma tarefa precedente. As restrições (3) garantem que cada máquina k, se usada, tenha uma sequencia exclusiva de tarefas. As restrições (4) definem que cada tarefa j tenha uma única tarefa imediata

sucessora, com exceção da tarefa 0 que estabelece o início e o fim de uma sequência de tarefas na máquina k. Para a tarefa 0, as restrições (5) estabelecem que os tempos de conclusão desta tarefa nas máquinas devem ser iguais a zero. Nas restrições (6) são verificados os instantes de conclusão das tarefas nas máquinas onde são executadas. Nas restrições (7) e (8) são definidos, respectivamente, o adiantamento e atraso associados a cada uma das tarefas. As restrições (9) e (10) indicam o domínio das variáveis utilizadas na formulação.

3 Heurísticas Propostas

A metaheurística GRASP, introduzida por Feo e Resende (1995), tem sido aplicada com êxito em variados problemas de otimização combinatória. Esta heurística consiste de um processo iterativo composto por duas fases, quais sejam: construção e busca local. Na primeira etapa, uma solução viável é construída utilizando um algoritmo guloso aleatorizado. Na etapa seguinte, a solução construída é submetida à etapa de busca local para ser aprimorada com objetivo de determinar um mínimo local. A construção e a busca local são aplicadas durantes um determinado número de iterações. Ao final das iterações, a melhor entre todas as solução encontradas é o resultado final da metaheurística.

Neste trabalho foi desenvolvida uma heurística GRASP para resolver o problema de sequenciamento de tarefas em MPNR. Com o objetivo de melhorar as soluções obtidas pela etapa de busca local, foi utilizado um procedimento de intensificação baseado na técnica *Path Relinking*. Nas subseções seguintes são descritos a forma da representação de uma solução, avaliação das soluções e cada uma das etapas da heurística GRASP, incluindo o procedimento de intensificação.

3.1 Representação e avaliação de soluções

Uma solução do problema de sequenciamento em MPNR é representada por uma lista encadeada l com m subsequências, sendo o resultado desta lista correspondente a uma solução s. Cada subsequência corresponde ao conjunto de tarefas a serem executadas em uma máquina. Um exemplo de uma solução para uma instância com oito tarefas e três máquinas seria $s = [[6, 4, 2], [7, 8, 1], [5, 3]]$, onde $[6, 4, 2]$, $[7, 8, 1]$ e $[5, 3]$ representam as subsequências de tarefas a serem executadas nas máquina 1, 2 e 3, respectivamente.

Para calcular o valor da função objetivo (1) para uma dada solução s é necessário determinar o tempo de conclusão da cada tarefa. Para isso, foi utilizado um algoritmo polinomial que efetua o cálculo dos tempos de conclusão ótimos. Na literatura, este algoritmo foi proposto para o caso de uma máquina por Davis e Kanet (1993), Lee e Choi (1995) e Wan e Yen (2002). Neste trabalho é utilizada uma adaptação para o caso de máquinas paralelas. O algoritmo é aplicado a cada uma das m subsequências com o objetivo de arranjar as tarefas, miminizando principalmente os adiantamentos. Na Figura 1 ilustra-se um exemplo do funcionamento do algoritmo aplicado para arranjar as quatro tarefas de uma subsequência correspondente a uma máquina. Na Figura 1 (a) mostra-se o processamento consecutivo das tarefas iniciando no tempo zero. Já na Figura 1 (b), as tarefas são arrumadas de tal maneira que elas finalizem o mais próximo possível da suas datas de entrega (d_i) e, neste caso são considerados períodos de ociosidades da máquina. Neste exemplo não são considerados os tempos de preparação ($setup$).

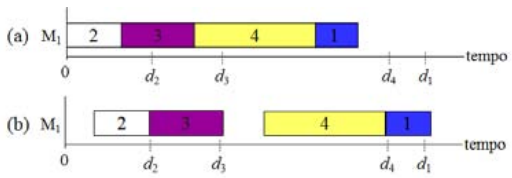


Figura 1: Exemplo de posicionamento das tarefas em uma máquina

3.1.1 Heurística de Construção GRASP

A etapa de construção da metaheurística GRASP é um processo iterativo, que tem como objetivo gerar uma solução viável. Iniciando com uma solução vazia s, a cada iteração da etapa de construção, uma tarefa é selecionada a partir de uma lista de candidatos (LC) e inserida na solução. Esta lista de candidatos é composta por todas as tarefas ainda não inseridas em s. A cada iteração, a lista LC das tarefas é ordenada considerando o incremento na função objetivo se a tarefa for inserida em s. Este processo é considerado adaptativo, pois o incremento na função objetivo associada a cada tarefa é atualizado para refletir a inserção realizada na iteração anterior. Este procedimento é detalhado no Algoritmo 1.

Algoritmo 1 Heurística de Construção (α)

$LC \leftarrow \{j_1, ..., j_n\}$; //Lista de candidatos (tarefas)
$s \leftarrow \emptyset$;
enquanto ($|LC| > 0$) **faça**
  Atualizar avaliação candidatos (LC);
  Ordenar (LC);
  $LRC \leftarrow$ Seleciona $\max(1, \alpha \times |LC|)$ tarefas melhor avaliadas de LC;
  $j_i \leftarrow$ Seleção Aleatória de uma tarefa de LRC;
  $s \leftarrow$Inserir(j_i, s);
  $LC \leftarrow LC - \{j_i\}$;
fim enquanto
retorne s;

As tarefas da lista LC são ordenadas em ordem não decrescente de acordo com o incremento no valor da função objetivo em relação a solução parcial s. Para calcular o incremento associado à tarefa j, para cada máquina k é inserida a tarefa j como última tarefa. Em seguida, os tempos ótimos de conclusão das tarefas é obtido e, então, é calculado o valor da função objetivo da solução parcial.

Este modo de inserir tarefas também é usado na heurística construtiva DJASA (*Dynamic Job Assignment with Setups Resource Assignment*) de Ruiz e Andrés (2007). A heurística DJASA é puramente gulosa, ou seja, a cada iteração sempre é adicionada a primeira tarefa da lista LC. No algoritmo de construção implementado neste trabalho, a tarefa adicionada é escolhida aleatoriamente de uma lista restrita de candidatos (LRC).

O tamanho da LRC, h, depende do valor do parâmetro α que indica o grau de aleatoriedade da heurística GRASP, onde $\alpha \in \Re$ e $\alpha \in [0, 1]$. Se $\alpha = 0$, então a LRC é composta apenas pelo melhor individuo da LC, o que faz a etapa de construção gulosa. No entanto, se $\alpha = 1$, então a LRC será igual a LC, tornando a etapa de construção aleatória.

Após a inserção de uma tarefa na solução em construção, a lista LC deve ser reordenada. Este processo é repetido até todas as tarefas de LC serem inseridas na solução, resultando numa solução completa.

3.1.2 Busca Local

A etapa de busca local da heurística GRASP é um procedimento iterativo que consiste em melhorar a solução s gerada pela etapa de construção. A busca local determina soluções vizinhas de s (ou seja, a vizinhança de s) realizando algumas alterações (movimentos) na sua estrutura. Dentre as vizinhas, escolhe-se aquela de melhor qualidade s'. Se o vizinho escolhido s' for melhor que s, a busca continua a partir de s' (ou seja, $s \leftarrow s'$). O procedimento finaliza quando não é possível melhorar a solução atual s, ou seja, quando esta solução for um ótimo local.

Para o Problema de Sequenciamento de Tarefas num ambiente com MPNR, soluções vizinhas são obtidas a partir da remoção de uma tarefa de sua posição original e posterior inserção em outra posição, na mesma máquina ou em outra máquina. Este movimento gera uma vizinhança de tamanho $(n^2 - n + m)$, se m for par, ou $(n^2 - m)$, se m for ímpar.

Por exemplo, para uma instância considerando 12 tarefas e 3 máquinas, como apresentado na Figura 2, para a solução $s = [[2, 5, 1, 4], [6, 11, 9, 10], [8, 7, 12, 3]]$, as soluções s' e s'' são consideradas duas soluções pertencentes à vizinhança de s. Para obter estas soluções a tarefa 4 foi removida de sua posição atual e reinserida na solução. Para $s' = [[2, 1, 4, 5], [6, 11, 9, 10], [8, 7, 12, 3]]$, a tarefa 4 foi recolocada em uma posição diferente ainda na máquina 1, enquanto que para $s'' = [[2, 1, 5], [6, 4, 11, 9, 10], [8, 7, 12, 3]]$ a tarefa 4 foi inserida na segunda posição da sequência de tarefas associadas à máquina 2.

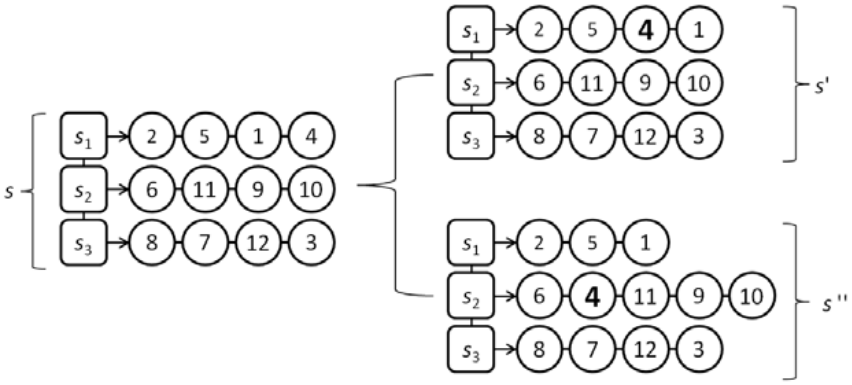


Figura 2: Exemplo de soluções vizinhas de s

3.2 Path Relinking

O *Path Realinking* (PR), originalmente introduzido por Glover (1996) no contexto da *Busca Tabu*, é uma estratégia de intensificação para explorar trajetórias que conectam duas soluções ao longo do espaço de soluções, na busca de soluções ainda melhores. Considerando duas soluções s_o e s_g, um caminho é gerado por meio da seleção de movimentos que introduzem, na solução base s_o, atributos da solução guia s_g. A cada etapa, uma vizinhança restrita da solução base é explorada, sendo selecionada a melhor dentre as soluções analisadas para assumir o papel de s_o e continuar o processo. Essa vizinhança restrita de s_o, $N_g(s_o)$, é composta por todas as soluções na vizinhança que incorporam um atributo da solução guia. Neste trabalho, as soluções de $N_g(s_o)$ são construídas utilizado movimentos de troca e inserção de tarefas da solução s_o.

No movimento de troca, apenas as tarefas de s_o que estão em posições diferentes em relação à posição em que aparecem na solução s_g são trocadas. Este movimento é utilizado para colocar estas tarefas nas posições corretas. A troca é realizada entre uma tarefa i de s_o com a tarefa que ocupa a posição de i na solução s_g.

O movimento de inserção é feito para que o número de tarefas em cada máquina da solução origem seja igual ao número de tarefas nas respectivas máquinas na solução guia. Este movimento remove a última tarefa de uma máquina de s_o que possui uma quantidade maior de tarefas comparado à mesma máquina em s_g. A tarefa removida é inserida na última posição de uma máquina com menor número de tarefas.

Na Figura 3 ilustra-se a geração da vizinhança restrita na aplicação do PR da solução $s_o = [[9, 6, 7, 8, 3], [12, 1, 10, 4], [2, 5, 11]]$ para $s_g = [[9, 6, 2, 8], [3, 12, 10, 1, 4], [7, 5, 11]]$. Neste exemplo, seis tarefas de s_o estão em posições distintas em relação a s_g, são elas 7, 3, 12, 1, 4 e 2. Assim, serão realizadas no máximo seis trocas. No exemplo, duas trocas não foram realizadas, relativas às tarefas 4 e 2. A tarefa 4 deveria ser trocada com a tarefa que ocupa a quinta posição na máquina 2, uma troca impossível já que a máquina não possui esta posição. Em relação à tarefa 2 da terceira máquina, a troca deveria ser com a tarefa 7 da máquina 1, um movimento desnecessário pois gera-se uma solução já analisada. Observa-se que a tarefa 3 da máquina 1 foi inserida na máquina 2, isso acontece pois estas máquinas possuem uma quantidade distinta de tarefas em s_o e s_g.

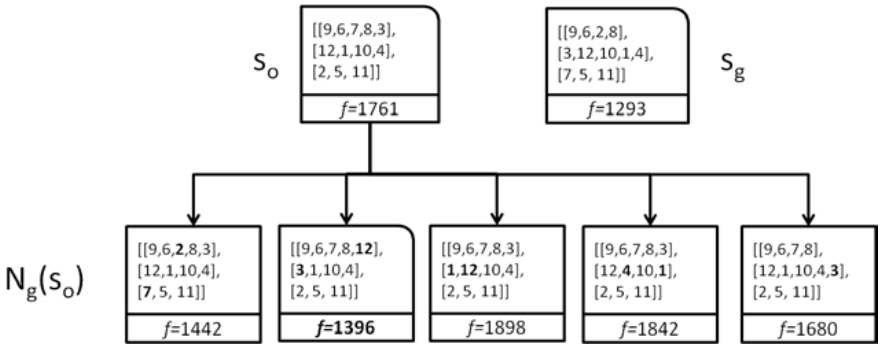


Figura 3: Exemplo do primeiro passo do *Path Relinking* para uma instância com $n = 12$ e $m = 3$

Neste trabalho foi adotada a estratégia *Mixed Path Relinking* descrita no trabalho de Resende e Ribeiro (2005). Nesta estratégia, entre as soluções s_o e s_g, dois caminhos são construídos simultaneamente, de s_o para s_g e de s_g para s_o, resultando na exploração de um número maior de soluções nas vizinhanças das duas soluções.

Para possibilitar a aplicação da estratégia de PR combinada a heurística GRASP foi utilizado um repositório de soluções, chamado de conjunto de soluções elite. Esse repositório armazena as soluções de melhor qualidade encontradas durante as iterações da metaheurística. Assim, como em Resende e Ribeiro (2005), a implementação de PR utiliza um conjunto E de soluções elites diferentes, de tamanho máximo E_{size}.

A atualização do conjunto elite é da seguinte forma. Se $|E| < E_{size}$, então uma nova solução s será inserida em E, caso $s \notin E$. Se $|E| = E_{size}$, a solução $s \notin E$ será inserida se ela for melhor que alguma solução deste conjunto. Dentre as soluções piores, aquela que tem menor diferença em relação a s será substituída, mantendo assim a diversidade do conjunto. A diferença entre duas soluções é determinada pelo número de tarefas que ocupam posições distintas.

O pseudocódigo da heurística híbrida GRASP+PR e apresentado no Algoritmo 2.

Algoritmo 2 GRASP+PR(α)

```
E ← ∅; //Conjunto Elite
f(s*) ← ∞;
i ← 1;
enquanto critério de parada não atingido faça
  s ← Heurística de Construção (α);
  s ← Local Search (s);
  se (i ≥ 2) então
    s' ← Seleção usando Roleta (E);
    s ← Path Relinking (s, s');
    Atualizar (E, s);
  fim se
  se f(s*) > f(s) então
    s* ← s
  fim se
  i ← i + 1
fim enquanto
return s*
```

O procedimento de PR implemento neste trabalho é executado a cada iteração do GRASP. O algoritmo recebe como parâmetros a solução s obtida pela heurística de busca local e uma solução s' selecionada através do método da roleta entre as soluções pertencentes ao conjunto elite E. A chance de uma solução $s_i \in E$ ser escolhida no método da roleta é proporcional a diferença entre essa solução s_i em relação a obtida pela busca local s.

4 Teste Computacionais

Neste trabalho testa-se a eficiência do PR quando incorporado na heurística GRASP. Assim, dois algoritmos heurísticos são avaliados, o GRASP (versão que não faz uso do PR) e GRASP+PR. Os algoritmos foram implementados em Java 1.6 e compilados com IDE Eclipse 3.5.1. Os testes foram efetuados em um computador dotado de um processador Intel Core2 Quad CPU Q9550 @ 2.83GHz, com 6 GB de RAM e utilizando o sistema operacional Windows 7.

Os teste computacionais foram divididos em três experimentos. No primeiro experimento é feita uma comparação entre os resultados das heurísticas implementadas e as soluções obtidas pelo software de otimização IBM ILOG CPLEX 12.0 através da resolução do modelo de PLIM. No segundo, as heurísticas GRASP e GRASP+PR são testadas usando um conjunto de problemas maiores. No terceiro é avaliado a probabilidade de distribuição do tempo de processamento das heurísticas. Esta avaliação utiliza *time-to-target plots* (Aiex *et al.*, 2002) para demostrar as diferenças entre desempenho de cada heurística.

Os parâmetros do GRASP e GRASP+PR foram definidos de forma experimental. A condição de parada para os algoritmos é baseada num tempo máximo de CPU definido por $(n \times m \times 50)$ milissegundos. O parâmetro usado na etapa de construção da heurística GRASP é fixado em $\alpha = 0.1$. O tamanho do conjunto de elite E, usado pela técnica *Path Relinking*, é $E_{size} = n/15$.

Para avaliar os resultados dos métodos propostos é computada a média dos Desvio Percentual Relativo (DPR). O DPR é calculado da seguinte maneira:

$$DPR = \frac{Metodo_{sol} - Melhor_{sol}}{Melhor_{sol}} * 100$$

onde $Metodo_{sol}$ é o valor da função objetivo obtido pelo método testado e $Melhor_{sol}$ é o melhor valor encontrado considerando todos os experimentos. As heurísticas foram executadas cinco vezes para cada instância e, neste caso, considera-se $Metodo_{sol}$ como a média do valor da função objetivo.

4.1 Geração de instâncias do problema

As instâncias dos problemas utilizados nos testes computacionais foram geradas a partir do trabalho de Lee e Pindedo (1997). Ao todo foram geradas 400 instâncias de diferentes tamanhos. As instâncias são classificados em 2 conjuntos de acordo às dimensões do problema: pequeno-médio e grande porte. O primeiro conjunto contém instâncias com $n \in \{8, 9, 10, 20, 30\}$ e $m \in \{3, 5\}$. Para cada combinação de n e m foram geradas 10 instâncias, totalizando 100 instâncias de pequeno-médio porte $(5 \times 2 \times 10 = 100)$. O conjunto de instâncias de grande porte é composto de 300 instâncias sendo que $n \in \{50, 60, 70, 80, 90, 100\}$ e $m \in \{10, 15, 20, 25, 30\}$. Também, para cada combinação de n e m foram geradas 10 instâncias $(6 \times 5 \times 10 = 300)$.

Para gerar as instâncias dos problemas são utilizados três parâmetros: o fator de dispersão das datas de entrega R, o fator de aperto das datas de entrega τ e o rigor dos tempos de preparação (*setups*) η. Todas as instâncias foram geradas utilizando $\tau = 0,3$, $R = 0,25$ e $\eta = 0,25$, da mesma maneira como em Lee e Pindedo (1997). O tempo de processamento da tarefa i na máquina k, $p_{i,k}$, é escolhido aleatoriamente do intervalo $[50, 100]$. As penalidades por atraso α_i e adiantamento β_i são escolhidas aleatoriamente no intervalo $[1, 100]$. Os tempos de preparação são gerados no intervalo $\left[\frac{2}{3}\eta\overline{p}, \frac{4}{3}\eta\overline{p}\right]$, sendo $\overline{p}$ a média dos tempos de processamento. Para gerar as datas de entrega é necessário determinar o intervalo no qual estas serão geradas. Para tal, considernado $\overline{v}$ o valor da média dos tempos de preparação, determina-se a estimativa do *makespam*, C_{max}, que é calculada pela seguinte fórmula:

$$C_{max} = \frac{n}{m}\left(\overline{p} + \overline{v}\left(0,4 + \frac{10m^2}{n^2} - \frac{\eta}{7}\right)\right)$$

As datas de entrega estão distribuídas uniformemente no intervalo $[(1 - R)\overline{d}, \overline{d}]$ com probabilidade τ e no intervalo $[\overline{d}, ((C_{max} - \overline{d})R) + \overline{d}]$ com probabilidade $(1 - \tau)$, onde $\overline{d} = C_{max}(1 - \tau)$ é a mediana das datas de entrega.

4.2 Experimento 1

Neste experimento são utilizados os problemas de pequeno é médio porte. Estes problemas foram resolvidos pelo software CPLEX limitando o tempo de execução em três horas. Na Tabela 1, na primeira coluna são apresentados os tamanhos das instâncias ($n \times m$), nas próximas três colunas são apresentados os DPR % para os três métodos comparados (CPLEX, GRASP e GRASP+PR) e nas duas últimas colunas são apresentados os tempos gastos pelas heurísticas e pelo CPLEX.

Para as instâncias com 8, 9 e 10 tarefas o CPLEX foi capaz de determinar a solução ótima num tempo inferior a 13 segundos. O GRASP+PR não encontrou as soluções ótimas para apenas algumas instâncas do grupo $n \times m = 9 \times 5$. Para as instâncias com 20 e 30 tarefas, os melhores resultados foram obtidos pela heurísticas GRASP+PR. O CPLEX, no tempo limite estabelecido, obteve soluções em média 56,8% piores com relação às melhores soluções heurísticas.

A partir dos resultados apresentados na Tabela 1 é possível verificar que a heurística GRASP, com ou sem o procedimento de intensificação (*Path Relinking*), foi capaz de obter soluções de melhor qualidade do que o CPLEX. A versão GRASP+PR obteve resultados em média $0,5\%$ melhores com relação à versão sem intensificação.

Os resultados do CPLEX para as instâncias médio porte sugerem que, o uso de heurísticas deve ser o mais apropriado para resolver problemas maiores.

Tabela 1: Média de Desvio Percentual Relativo (DPR %) - instâncias de pequeno e médio porte

$n \times m$	DPR %			Tempo (s)	
	CPLEX	GRASP	GRASP+PR	Heurísticas	CPLEX
8×3	0,00	0,00	0,00	1,20	8,47
8×5	0,00	0,18	0,00	2,00	8,03
9×3	0,00	0,61	0,00	1,35	9,44
9×5	0,00	0,25	0,08	2,25	12,54
10×3	0,00	0,14	0,00	1,50	7,54
10×5	0,00	1,16	1,04	2,50	6,54
20×3	19,38	0,76	0,19	3,00	10.800,00
20×5	29,63	2,22	0,78	5,00	10.800,00
30×3	66,60	1,42	0,98	4,50	10.800,00
30×5	111,54	2,29	0,94	7,50	10.800,00
Média	22,72	0,90	0,40	3,08	4.325,26

4.3 Experimento 2

No segundo experimento, as heurísticas GRASP e GRASP+PR foram testadas nas 300 instâncias de grande porte. Na Tabela 2, na primeira coluna são apresentados os tamanhos das instâncias ($n \times m$) e nas colunas seguintes são apresentados os valores do DPR% das heurísticas. Observa-se que, para todas as instâncias resolvidas, a versão GRASP+PR obteve menores DPRs, o que mostra que as soluções encontradas por esta heurística são de melhor qualidade. Utilizando *Path Relinking*, os resultados da heurística GRASP melhoraram em média 3.8%.

Na Figura 4 mostra-se o *boxplot* das distribuições de médias do DPR correspondente às heurísticas comparadas. Nesta figura observa-se que a heurística GRASP+PR apresenta uma média de DPR menor que a média obtida pelo GRASP. Esta figura também mostra que a heurística híbrida GRASP+PR possui a menor dispersão da média do DPR.

Para validar os resultados das heurísticas é interessante verificar se as diferenças entre as médias de DPR são estatisticamente significativas. Os testes estatísticos foram feitos utilizando o *software* R 2.13 com um nível de significância de 95%, que determina a confiabilidade do teste. Foi realizada um teste *t* para testar a significância estatística das diferenças entre as médias do DPR através da comprovação ou rejeição de uma hipótese nula (não existe diferença entre as médias do DPR das heurísticas). O resultado do teste *t* é apresentado na Tabela 3. As colunas *t*, *df* e

Tabela 2: Média de Desvio Percentual Relativo (DPR %) (instâncias de grande porte)

Problema	GRASP	GRASP + PR	Problema	GRASP	GRASP + PR
50 x 10	6,04	2,88	80 x 10	4,83	3,58
50 x 15	8,87	3,92	80 x 15	6,21	4,03
50 x 20	10,78	5,63	80 x 20	8,21	4,76
50 x 25	14,15	7,15	80 x 25	9,78	4,97
50 x 30	13,12	6,97	80 x 30	10,14	5,28
60 x 10	5,72	3,46	90 x 10	5,39	3,88
60 x 15	8,08	3,84	90 x 15	6,53	4,65
60 x 20	7,94	2,57	90 x 20	8,66	5,27
60 x 25	10,17	4,97	90 x 25	8,25	4,48
60 x 30	15,70	8,10	90 x 30	9,21	4,61
70 x 10	5,55	3,77	100 x 10	5,40	4,59
70 x 15	6,07	3,22	100 x 15	4,63	3,03
70 x 20	8,91	5,09	100 x 20	7,70	4,65
70 x 25	10,08	4,54	100 x 25	8,16	4,60
70 x 30	10,65	5,26	100 x 30	8,71	4,86
Média	9,45	4,76	Média	7,45	4,48

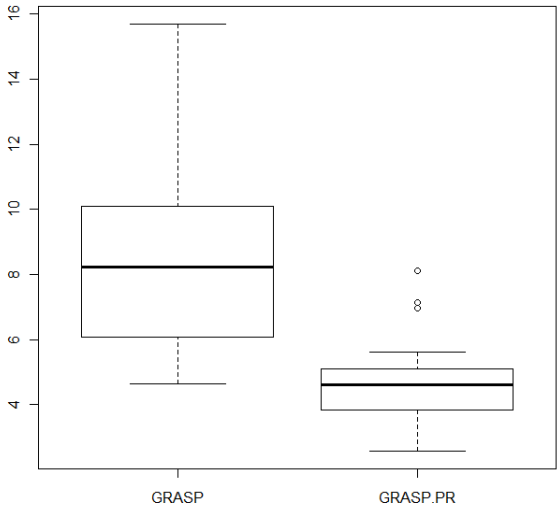


Figura 4: Boxplot da média de resultados do conjunto grande

p-value representam, respectivamente, o valor de *t-Student*, o grau de liberdade e a probabilidade de aceitação da hipótese nula. Como o *p-value* é menor $0,05$ a hipótese nula será rejeitada e é possível afirmar que a diferença entre os resultados é estatisticamente significante, ou seja, a heurística híbrida GRASP+PR é significativamente melhor que a heurística GRASP.

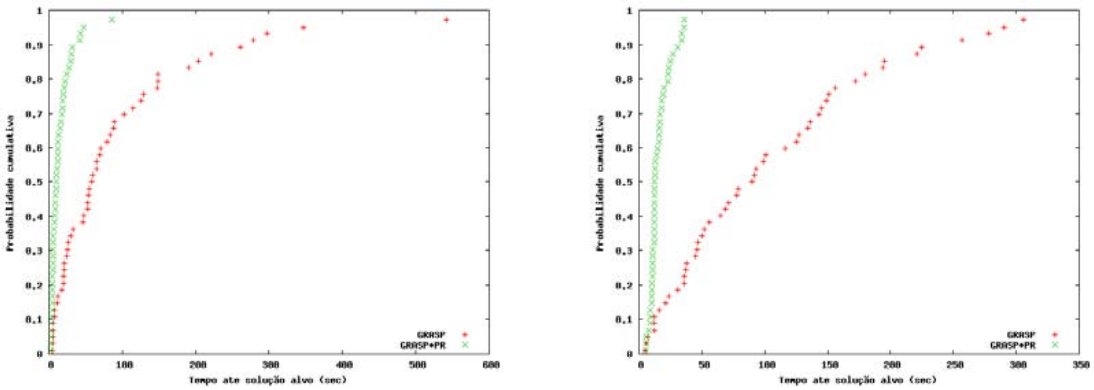
Tabela 3: Resultados estatísticos da média do desvio percentual relativo (DPR)

t	*df*	*p-value*
7.072	40.482	$1.396e-08$

4.4 Experimento 3

O propósito deste experimento é analisar a convergência das heurísticas propostas. Cada heurística (GRASP, GRASP+PR) foi executada 50 vezes. A condição de parada considerada é atingir um valor alvo. A solução alvo considerada é a melhor solução obtida pelo algoritmo GRASP depois de $n \times m \times 50$ milissegundos de tempo de CPU. Neste experimento são feitos gráficos dos

testes de probabilidade empírica como proposto por Aiex *et al*. (2002). Na Figura 5 são ilustrados os testes para duas instâncias de grande porte (30×50 e 30×80). Estas figuras mostram que a heurística híbrida GRASP+PR possui uma maior probabilidade para encontrar a solução alvo gastando o menor tempo de CPU. Dos resultados obtidos, pode-se concluir que a hibridização da heurística GRASP com PR foi vantajosa para o melhor desempenho da mesma.



(a) Problema com 50 tarefas e 30 maquinas (b) Problema com 80 tarefas e 30 maquinas

Figura 5: Testes de probabilidade empírica para instancias de com 50 e 80 tarefas.

5 Conclusão

Neste trabalho foi abordado o problema de sequenciamento de tarefas em máquinas paralelas não relacionadas considerando tempo de preparação de máquina com o objetivo de minimizar a soma de penalidades por adiantamento e atraso da produção. Para tratar o problema foi proposta uma heurística híbrida GRASP com *Path Relinking*. Para o *Path Relinking*, foi utilizado a estratégia *mixed* no qual dois caminhos são explorados simultaneamente.

As heuríticas GRASP e GRASP+PR foram testadas utilizando dois conjuntos de instâncias, pequeno-médio e grande porte. Com o uso do CPLEX foi possível obter as soluções ótimas apenas para as instâncias com até 10 tarefas. Para as instâncias com mais de 20 tarefas a utilização do CPLEX é inviável e, neste caso, os métodos mais adequados para resolvê-las são as heurísticas.

A partir dos experimentos computacionais e análise estatística realizada é possível concluir que a utilização da intensificação com *Path Relinking* proporcionou uma melhoria significativa na heurística GRASP. Como trabalho futuro sugere-se a inclusão de outros procedimentos de aprimoramento, tais como *Iterated Local Search*, de forma a melhorar os resultados obtidos.

Referências

Aiex, R., Resende, M. e Ribeiro, C. (2002), Probability distribution of solution time in GRASP: An experimental investigation. *J. of Heuristics*, v. 8, p. 343–373.

Arenales, M., Morabito, R., Armentano, V. A. e Yanasse, H. *Pesquisa operacional*. Elsevier, Rio de Janeiro, 2007.

Armacost, R. L. e Salem, A. Unrelated parallel machine scheduling with setup times and machine eligibility constrints. *Proceedings of the Eighth Industrial Engineering Research Conference*, 1999.

Cheng, T. e Sin, C. (1990), A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, v. 9, p. 271–292.

Davis, J. e Kanet, J. (1993), Single machine scheduling with early and tardy completion costs. *Naval Research Logistics*, v. 40, p. 85–101.

Feo, T. e Resende, M. (1995), Greedy Randomized Adaptive Search Procedures. *J. of Global Optimization*, v. 6, p. 109–133.

Festa, P. e Resende, M. (2009), An annotated bibliography of GRASP, Part II: Applications. *International Transactions in Operational Research*, v. 16, p. 131–172.

Fowler, J., Horng, S. e Cochran, J. (2003), A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups. *Industrial Engineering: Theory Applications and Practice*, v. 10, p. 232–243.

França Filho, M. *GRASP e Busca Tabu aplicados a problemas de programação de tarefas em máquinas paralelas*. Tese de doutorado, Engenharia de Sistemas, UNICAMP, 2007.

Glover, F. (1996), Tabu search and adaptive memory programing. *Interface in Computer Science and Operational Reaserch*, v. , p. 1–75.

Guo, Y., Lim, A., Rodrigues, B. e Yang., L. (2007), Minimizing the makespan for unrelated parallel machines. *nternational Journal of Artificial Intelligence Tools*, v. 16, p. 399–415.

Lee, C. Y. e Choi, J. Y. (1995), A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers & Operations Research*, v. 22, p. 857–869.

Lee, Y. H. e Pindedo, M. (1997), Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, v. 100, p. 464–474.

Liaee, M. e Emmons, H. (1997), Scheduling families of jobs with setup times. *International Journal of Production Economics*, v. 51, p. 165–176.

Logendran, R. e Saputra, H. Real-time scheduling of jobs with sequence-dependent setups on unrelated parallel machines. *Proceedings of the Eighth Industrial Engineering Research Conference - IERC*, Phoenix, AZ, 1999.

Potts, C. e Kovalyov, M. (2000), Scheduling with batching: a review. *European Journal of Operational Research*, v. 120, p. 228–249.

Resende, M. e Ribeiro, C. GRASP with path-relinking: Recent advances and applications. Ibaraki, T., Nonobe, K. e Yagiura, M. (Eds.), *Metaheuristics: Progress as Real Problem Solvers*, p. 29–63. Springer, 2005.

Ruiz, R. e Andrés, C. Unrelated parallel machines scheduling with resource-assignable sequence dependent setup times. *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling Theory and Applications*, p. 439–446, 2007.

Tamimi, S. e Rajan, V. Reduction of total weighted tardiness on uniform machines with sequence dependent setups. *Industrial Engineering Research*, 1997.

Vallada, E. e Ruiz, R. (2011), Genetic algorithms for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, v. 3, p. 612–622.

Wan, G. e 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*, v. 142, p. 129–146.