UMA HEURÍSTICA DE REDUÇÃO DO ESPAÇO DE BUSCA PARA UMA CLASSE DE PROBLEMAS DE SEQUENCIAMENTO DE TAREFAS EM UMA MÁQUINA

Bruno Ferreira Rosa*, Marcone Jamilson Freitas Souza†, Sérgio Ricardo de Souza*

*Centro Federal de Educação Tecnológica de Minas Gerais
Avenida Amazonas, 7675, CEP 30.510-000
Belo Horizonte, Minas Gerais, Brasil

†Universidade Federal de Ouro Preto
Departamento de Computação, Campus Universitário, CEP 35.400-000
Ouro Preto, Minas Gerais, Brasil

Emails: brunofazmat@dppg.cefetmg.br, marcone@iceb.ufop.br, sergio@dppg.cefetmg.br

Abstract— This work deals with the problem of job scheduling in a single-machine. In the addressed problem, the setup times of the machine depends on the production sequence and each job has a processing time and a time window within which it should preferably be completed. The aim is to minimize the weighted sum of the tardiness and earliness as they are working. In this work, it is proposed a heuristic algorithm composed of two phases to solve the problem. First, it generates a solution based on GRASP, Proximate Optimality Principle and Variable Neighborhood Descent (VND). Second, post-refinement of this solution is made by the application of the another local search based on VND. In order to reduce the computational cost it was also proposed a strategy to reduce the search space. The heuristic algorithm proposed here proved to be better than other proposals in the literature, having the low computational cost as their major ally. Thus, it was possible to resolve, in feasible computing time, problems of greater dimensions than the ones solved until now.

Keywords— Combinatorial Optimization, Heuristics, Single-machine scheduling.

Resumo— Este trabalho trata do problema de sequenciamento de tarefas em uma máquina. No problema abordado, os tempos de preparação da máquina são dependentes da sequência de produção e cada tarefa está associada a um tempo de processamento e a uma janela de tempo, dentro da qual ela deve ser preferencialmente concluída. O objetivo é minimizar a soma ponderada dos atrasos e das antecipações na execução de tais tarefas. Propõe-se, neste trabalho, um algoritmo heurístico composto de duas fases para resolvê-lo. Na primeira, gera-se uma solução com base na metaheurística GRASP, no Princípio da Otimalidade Próxima e na Descida em Vizinhança Variável (VND). Na segunda, faz-se o pós-refinamento dessa solução por meio da aplicação de outra busca local baseada em VND. Com o objetivo de reduzir o custo computacional, adicionalmente é proposta uma estratégia de redução do espaço de busca. O algoritmo heurístico proposto mostrou-se melhor que outras propostas apresentadas na literatura, tendo o baixo custo computacional como o seu grande aliado. Dessa forma, foi possível resolver, em tempo computacional viável, problemas de dimensões maiores que os resolvidos até então.

Palavras-chave— Otimização combinatória, Heurísticas, Sequenciamento em uma máquina.

1 Introdução

O planejamento criterioso das atividades produtivas é importante na redução dos custos de produção. Segundo França Filho (2007), concluir uma tarefa antecipadamente pode resultar em custos financeiros extras pela necessidade de disponibilização antecipada de capital, necessidade de espaço para armazenamento ou necessidade de outros recursos para manter e gerenciar o estoque. Do mesmo modo, concluir uma tarefa com atraso pode resultar em multas contratuais, perda de credibilidade da empresa e redução de vendas. Assim, a redução desses tempos de antecipação ou atraso pode acarretar em uma significativa diminuição dos custos do processo.

De acordo com Baker and Scudder (1990), o Problema de Sequenciamento em Uma Máquina com Penalidades por Antecipação e Atraso da produção (PSUMAA) modela bem ambientes de produção administrados com o objetivo acima. O PSUMAA consiste em sequenciar e determinar o momento em que as tarefas devem ser executadas em uma máquina, com o objetivo de minimizar a soma ponderada das antecipações e dos atrasos na produção de tais tarefas. Trata-se de um problema NP-difícil (Baker and Scudder, 1990) e com muitas aplicações práticas em indústrias metalúrgicas, têxteis, de tintas, entre outras.

As datas de entrega das tarefas podem ser comuns ou distintas. Uma abordagem mais genérica e menos explorada na literatura considera janelas de entrega distintas, em que cada tarefa deve ser preferencialmente concluída dentro de um respectivo período de tempo. De acordo com Koulamas (1996), esse último caso ocorre quando existem tolerâncias em torno das datas desejadas para a entrega de cada tarefa. Sendo assim, as tarefas concluídas dentro de suas respectivas janelas de entrega não incorrem em penalidades, ao contrário daquelas concluídas fora dessas janelas.

Nas indústrias em que são produzidos diferentes tipos de produtos e em que existe uma troca frequente do tipo de tarefa executada em uma máquina, geralmente é necessário, após a conclusão de uma tarefa, preparar a máquina antes do

início da execução da tarefa seguinte (Allahverdi et al., 1999). Este tempo de preparação, chamado tempo de *setup*, inclui os tempos gastos para trocar as ferramentas, preparar o material, limpar a máquina etc. Muitos trabalhos em problemas de sequenciamento assumem que os tempos de preparação são independentes da sequência de produção, isto é, que eles são desprezíveis ou podem ser acrescentados aos tempos de processamento das tarefas. No entanto, pesquisas mostram que, em grande parte das situações práticas, tais tempos são dependentes da sequência de produção e devem ser considerados.

Considerações sobre a continuidade do funcionamento da máquina também podem ser impostas ao PSUMAA. Conforme França Filho (2007), existem situações em que a ociosidade das máquinas não é permitida por implicar em custos mais elevados que aqueles decorrentes da conclusão antecipada das tarefas. No entanto, o mesmo autor chama a atenção para o fato de que há casos em que manter a máquina inativa é vantajoso, ainda que exista uma tarefa disponível para processamento.

Este trabalho aborda uma classe genérica do PSUMAA. No problema abordado, considera-se a existência de janelas de entrega distintas, tempos de preparação dependentes da sequência de produção e são permitidos tempos ociosos entre as execuções de tarefas consecutivas. Doravante, esse problema será denotado por PSUMAA-JT.

Dentre os trabalhos da literatura que tratam do PSUMAA-JT, destacam-se os de Gomes Júnior (2007), Penna (2009) e Ribeiro (2009). No primeiro trabalho foi proposto um modelo de programação matemática para representar o problema. Também foi proposto um método heurístico baseado em GRASP, *Iterated Local Search* (ILS) e Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND). Para cada sequência de tarefas gerada pela heurística proposta, é utilizado um algoritmo de complexidade polinomial para determinar a data ótima de início de processamento das tarefas daquela sequência. Foram realizados experimentos computacionais em problemas-teste com os tempos de preparação simétricos e com até 75 tarefas. O segundo trabalho propõe uma heurística baseada em GRASP, VND, Busca Tabu e Reconexão por Caminhos, enquanto no terceiro trabalho é proposto um Algoritmo Genético Adaptativo (AGA) para resolver o problema. Nos dois últimos trabalhos foram realizados experimentos nos mesmos problemas-teste utilizados por Gomes Júnior (2007), sendo mostrada a superioridade dos algoritmos propostos em relação a este último. Ribeiro (2009) também realizou testes em outras duas bases de dados em que os tempos de preparação não são necessariamente simétricos. Foram resolvidos problemas-teste com até 100 tarefas por meio do AGA proposto.

Apesar de já existirem na literatura trabalhos que utilizam técnicas heurísticas para resolver o PSUMAA-JT, os procedimentos utilizados são genéricos, ou seja, não fazem uso de propriedades ou características do problema para reduzir o espaço de busca e, consequentemente, reduzir também o tempo de processamento. O presente trabalho contribui com um algoritmo heurístico, baseado nos métodos GRASP (Feo and Resende, 1995), Princípio da Otimalidade Próxima - POP (Glover and Laguna, 1997) e Descida em Vizinhança Variável - VND (Mladenović and Hansen, 1997), para resolver o PSUMAA-JT. Com o objetivo de reduzir o custo computacional, também é proposta uma estratégia de redução do espaço de busca.

O restante deste trabalho está organizado como segue. Na Seção 2 é feita uma descrição detalhada do problema estudado e na Seção 3 é descrito o algoritmo heurístico desenvolvido para resolvê-lo. Na Seção 4 são apresentados e discutidos os resultados computacionais, enquanto a Seção 5 conclui o trabalho e sugere perspectivas futuras para o melhoramento do algoritmo proposto.

2 Descrição do problema

O PSUMAA-JT abordado neste trabalho possui as seguintes características:

- Uma máquina deve processar um conjunto I de n tarefas;
- Associado a cada tarefa $i \in I$ está:
  - um tempo de processamento P_i;
  - uma janela de entrega $[E_i, T_i]$, dentro da qual a tarefa i deve ser preferencialmente concluída;
  - um custo α_i por unidade de tempo de antecipação;
  - um custo β_i por unidade de tempo de atraso;
- Há antecipação de uma tarefa $i \in I$ quando seu processamento é concluído antes de E_i;
- Há atraso de uma tarefa $i \in I$ quando seu processamento é concluído depois de T_i;
- As tarefas que forem concluídas dentro de suas respectivas janelas de entrega não geram penalidades;
- A máquina executa no máximo uma tarefa por vez e, uma vez iniciado o processamento de uma tarefa, não é permitida a sua interrupção;
- Todas as tarefas estão disponíveis para processamento na data 0;
- Entre duas tarefas i e $j \in I$ consecutivas, é necessário um tempo S_{ij} de preparação da máquina. Assume-se que o tempo de preparação da máquina para o processamento da primeira tarefa na sequência é igual a 0;

- É permitido tempo ocioso entre a execução de duas tarefas consecutivas.

O objetivo é determinar uma sequência de produção e as datas de término de produção das tarefas de forma a minimizar a soma ponderada das antecipações e atrasos, ou seja, minimizar

$$Z = \sum_{i=1}^{n} (\alpha_i e_i + \beta_i t_i) \qquad (1)$$

para $e_i = \max\{0,\ E_i - C_i\}$ e $t_i = \max\{0,\ C_i - T_i\}$, sendo C_i a data de conclusão da tarefa i.

3 Algoritmo proposto

3.1 GPV

O algoritmo proposto, denotado por GPV e cujo pseudocódigo está apresentado na Figura 1, possui duas fases. Na primeira (linhas 3 a 12 da Figura 1), constrói-se uma solução de acordo com a fase de construção do GRASP (Feo and Resende, 1995) e com o Princípio da Otimalidade Próxima - POP (Glover and Laguna, 1997), usando-se a Descida em Vizinhança Variável - VND (Mladenović and Hansen, 1997) como procedimento de busca local do POP. Na segunda fase (linha 13 da Figura 1), faz-se o pós-refinamento da solução proveniente da fase anterior, por meio da aplicação de outra busca local baseada em VND. Os detalhes do GPV são discutidos nas subseções seguintes.

```
   Algoritmo GPV()
1  f* ← +∞;
2  Iter ← 0;
3  enquanto (Iter < GRASPmax) faça
4      Iter ← Iter + 1;
5      v0 ← ConstroiSolucaoPOP();
6      v ← VND1(v0);
7      se (f(v) < f*) então
8          v* ← v;
9          f* ← f(v);
10         Iter ← 0;
11     fim-se
12 fim-enquanto;
13 v* ← VND2(v*);
14 Retorne v*;
   fim GPV;
```

Figura 1: Algoritmo GPV

3.2 Representação de uma solução

Uma solução (sequência) para o PSUMAA-JT de n tarefas é representada por um vetor v de n posições, em que cada posição $i = 1,\ 2,\ \cdots,\ n$ indica a ordem de produção da tarefa v_i. Por exemplo, na sequência $v = (5,\ 3,\ 2,\ 1,\ 4,\ 6)$, a tarefa 5 é a primeira a ser executada e a tarefa 6 é a última.

3.3 Vizinhança de uma solução

Para explorar o espaço de soluções são usados três tipos de movimentos: troca da ordem de processamento de duas tarefas da sequência de produção, realocação de uma tarefa para outra posição da sequência e realocação de um bloco de tarefas para outra posição da sequência. Esses movimentos definem, respectivamente, as vizinhanças N^T, N^R e N^{RB}.

Na vizinhança N^T, um exemplo de vizinho da solução $v = (5,\ 3,\ 2,\ 1,\ 4,\ 6)$ é a solução $v' = (5,\ 4,\ 2,\ 1,\ 3,\ 6)$, pois v' é obtida por meio da troca da ordem de processamento da segunda com a quinta tarefa de v.

Em uma sequência com n tarefas, cada uma delas pode trocar de posição com qualquer uma das outras $n-1$ tarefas. Como trocar a ordem de processamento da i-ésima tarefa da sequência com a j-ésima tarefa é equivalente à troca da ordem de processamento da j-ésima tarefa na sequência com a i-ésima tarefa para todo $i, j \in \{1, 2, \cdots, n\}$, então há $n(n-1)/2$ vizinhos distintos na vizinhança N^T.

A solução $v' = (5,\ 2,\ 1,\ 4,\ 3,\ 6)$ é um exemplo de vizinho da solução $v = (5,\ 3,\ 2,\ 1,\ 4,\ 6)$ na vizinhança N^R, pois é obtida pela realocação da tarefa 3, que está na segunda posição em v, para a quinta posição na sequência de produção.

Dada uma sequência de n tarefas, cada uma delas pode ser realocada para $n-1$ posições distintas. Além disso, realocar a tarefa da i-ésima posição para a posição $i+1$ na sequência é equivalente a realocar a tarefa da posição $i+1$ para a i-ésima posição na sequência, para todo $i \in \{1,\ 2,\ \cdots,\ n-1\}$. Sendo assim, há $n(n-1) - (n-1) = (n-1)^2$ vizinhos distintos na vizinhança N^R.

Já na vizinhança N^{RB}, um exemplo de vizinho da solução $v = (5,\ 3,\ 2,\ 1,\ 4,\ 6)$ é a solução $v' = (1,\ 4,\ 5,\ 3,\ 2,\ 6)$, a qual é obtida pela realocação do bloco de três tarefas $<5,\ 3,\ 2>$, que está sequenciado no início de v, para depois da tarefa 4 na sequência de produção, ou seja, para duas posições sucessoras.

Em uma sequência com n tarefas existem $n-k+1$ blocos distintos de k tarefas, para todo $k \in \{1,\ 2,\ \cdots,\ n\}$, e cada um destes blocos pode ser realocado para $n-k$ posições distintas da sequência. Além disso, realocar um bloco de i tarefas para j posições sucessoras é equivalente à realocação de um bloco de j tarefas para i posições antecessoras, para todo $i,\ j \in \{1,\ 2,\ \cdots,\ n\}$ e $i + j \leq n$. Dessa forma, há $\left(\sum_{k=1}^{n}(n-k+1)(n-k)\right)/2 = \left(\sum_{k=1}^{n}(n-k)^2 + \sum_{k=1}^{n}(n-k)\right)/2 = = \left(n\,(2\,n-1)\,(n-1)/6 + n\,(n-1)/2\right)/2 = n\,(n-1)\,(n+1)/6$ vizinhos distintos na vizinhança N^{RB}.

Com o objetivo de evitar o retorno a vizinhos

já analisados e, consequentemente, reduzir o custo computacional, todas as observações mencionadas nos parágrafos anteriores foram consideradas na implementação computacional do algoritmo GPV.

3.4 Função de Avaliação

Como os movimentos utilizados não geram soluções inviáveis, uma sequência v é avaliada pela função f dada pela equação (1), a qual deve ser minimizada. Para determinar os valores de e_i e t_i para uma dada sequência de tarefas, é utilizado o algoritmo de determinação das datas ótimas de início de processamento das tarefas (ADDOIPT), proposto por Gomes Júnior (2007).

3.5 Construção de uma Solução

Nesta etapa do GPV, uma solução é formada, tarefa por tarefa, de forma parcialmente gulosa, seguindo as ideias da fase de construção do algoritmo GRASP e o POP. Sendo assim, a cada iteração desta etapa as tarefas que ainda não foram sequenciadas são avaliadas por meio de uma função adaptativa gulosa g que estima o benefício associado à inclusão de cada tarefa na subsequência corrente. As tarefas $t \in I$ tais que $g(t) \leq g_{min} + \gamma \times (g_{max} - g_{min})$, com $\gamma \in [0,1]$, são inseridas em uma lista restrita de candidatos (LRC). A partir de LRC, escolhe-se, aleatoriamente e com probabilidade uniforme, uma tarefa que será incrementada à solução parcial corrente.

O tamanho da LRC e o nível de gulosidade e de aleatoriedade da solução construída são determinados por γ. Quanto mais próximo de 0 estiver γ, maior o nível de gulosidade e menor o de aleatoriedade. Dessa forma, $\gamma = 0$ implica em uma solução completamente gulosa e $\gamma = 1$ em uma solução totalmente aleatória.

A Figura 2 mostra o pseudocódigo do procedimento proposto para construir uma solução.

```
   procedimento ConstroiSolucaoPOP()
1  s ← ∅;
2  Inicialize o conjunto C de tarefas;
3  enquanto (C ≠ ∅) faça
4      g_min ← min{g(t) | t ∈ C};
5      g_max ← max{g(t) | t ∈ C};
6      LRC ← {t ∈ C | g(t) ≤
       g_min + γ × (g_max − g_min)};
7      Escolha um elemento t ∈ LRC ;
8      s ← s ∪ {t};
9      s ← MDR(s) ;   /* Aplique o POP */
10     Atualize o conjunto C de candidatos;
11 fim-enquanto;
12 Retorne s;
   fim ConstroiSolucaoPOP;
```

Figura 2: Construção de solução com POP

Para estimar o benefício da inserção de cada tarefa t ainda não sequenciada na iteração i, em cada iteração em que o procedimento *ConstroiSolucaoPOP* é aplicado, são utilizadas seis funções adaptativas gulosas. Tais funções foram motivadas pelas características do problema e são apresentadas a seguir:

1. $g_1(t) = \frac{2E_t}{\max\{E_k \mid k \in F_i\}} + \frac{2T_t}{\max\{T_k \mid k \in F_i\}} + \frac{\alpha_t}{\max\{\alpha_k \mid k \in F_i\}} - \frac{\beta_t}{\max\{\beta_k \mid k \in F_i\}} + \frac{P_t}{\max\{P_k \mid k \in F_i\}} + \frac{S_{i-1,t}}{\max\{S_{i-1,k} \mid k \in F_i\}}$;
2. $g_2(t) = \frac{2E_t}{\max\{E_k \mid k \in F_i\}} + \frac{T_t}{\max\{T_k \mid k \in F_i\}} + \frac{\alpha_t}{\max\{\alpha_k \mid k \in F_i\}} - \frac{\beta_t}{\max\{\beta_k \mid k \in F_i\}} + \frac{P_t}{\max\{P_k \mid k \in F_i\}} + \frac{S_{i-1,t}}{\max\{S_{i-1,k} \mid k \in F_i\}}$;
3. $g_3(t) = \frac{E_t}{\max\{E_k \mid k \in F_i\}} + \frac{2T_t}{\max\{T_k \mid k \in F_i\}} + \frac{\alpha_t}{\max\{\alpha_k \mid k \in F_i\}} - \frac{\beta_t}{\max\{\beta_k \mid k \in F_i\}} + \frac{P_t}{\max\{P_k \mid k \in F_i\}} + \frac{S_{i-1,t}}{\max\{S_{i-1,k} \mid k \in F_i\}}$;
4. $g_4(t) = E_t + T_t$;
5. $g_5(t) = T_t$;
6. $g_6(t) = E_t$.

Nas funções g_1, g_2 e g_3, F_i representa o conjunto das tarefas ainda não sequenciadas até a i-ésima iteração. Para $i = 1$, em lugar de $\frac{S_{i-1,t}}{\max\{S_{i-1,k} \mid k \in F_i\}}$, é utilizado $-\frac{\text{média}\{S_{kt} \mid k \in I \text{ e } k \neq t\}}{\max\{S_{kl} \mid k,l \in I \text{ e } k \neq l\}}\Big)$.

Observa-se que as funções g_1, g_2 e g_3 consideram, simultaneamente, todas as características das tarefas, diferenciando-se apenas pelos pesos atribuídos às datas de início e de término das janelas de entrega. A função g_4 considera, simultaneamente, as datas de início e de término das janelas de entrega das tarefas. A função g_5 considera apenas as datas de término das janelas de entrega, enquanto a função g_6 leva em consideração apenas as datas de início das janelas de entrega.

Tais funções são utilizadas segundo a ordem apresentada anteriormente. Nas seis primeiras iterações, ou seja, na primeira vez em que cada função g é utilizada, faz-se $\gamma = 0$.

Ainda na etapa de construção de uma solução, é aplicado o POP (vide linha 9 da Figura 2). Este princípio é baseado na idéia de que boas soluções em um nível estão próximas de boas soluções em um nível adjacente (Glover and Laguna, 1997). Assim, sempre que uma nova tarefa é inserida na solução em formação, esta é submetida a uma busca local. A busca local utilizada consiste em uma descida randômica com a vizinhança N^R. Dada uma solução, aleatoriamente escolhe-se uma tarefa na sequência e uma nova posição para ela. Se a nova sequência produzir uma solução de melhora, a nova sequência é aceita e passa a ser a solução corrente; caso contrário, é testada outra realocação aleatória. A busca é interrompida após *MDRMax* realocações consecutivas sem melhora na solução corrente, sendo *MDRMax* um parâmetro do procedimento.

3.6 VND_1

Para refinar as soluções construídas na primeira fase do GPV (linha 6 da Figura 1), utiliza-se um VND, denotado por VND_1. Esta busca local consiste de três passos:

1. Descida randômica usando vizinhança N^T;
2. Descida randômica usando vizinhança N^R;
3. Descida randômica usando vizinhança N^{RB}.

No passo 1, duas tarefas são escolhidas aleatoriamente e a ordem de seus processamentos na sequência de produção é trocada. Se a nova sequência produzir uma solução de melhora, ela é aceita e passa a ser a solução corrente; caso contrário, é testada outra troca aleatória. O passo 1 termina quando ocorrer *VNDmax* trocas consecutivas sem melhora na solução corrente, sendo *VNDMax* um parâmetro do procedimento. Neste último caso, passa-se para o passo 2.

No passo 2, aleatoriamente escolhe-se uma tarefa na sequência e uma nova posição para ela. Se a nova sequência produzir uma solução de melhora, a nova sequência passa a ser a solução corrente e volta-se ao passo 1; caso contrário, é testada outra realocação aleatória. O passo 2 é interrompido após *VNDMax* realocações consecutivas sem melhora na solução corrente. Neste último caso, passa-se para o passo 3.

No passo 3, aleatoriamente escolhe-se um bloco de tarefas na sequência de produção (o tamanho do bloco também é escolhido de forma aleatória no intervalo $[1, n-1]$) e uma nova posição para ele. Se a nova sequência produzir uma solução de melhora, ela passa a ser a solução corrente e volta-se ao passo 1; caso contrário, é testada outra realocação de bloco aleatória. O passo 3 é interrompido após *VNDMax* realocações de blocos consecutivas sem melhora na solução corrente. Neste último caso, o VND_1 é encerrado e a melhor solução encontrada é retornada.

3.7 VND_2

A solução oriunda da primeira fase do GPV (linhas 1 a 12 da Figura 1) não é necessariamente um ótimo local em relação às vizinhanças adotadas, visto que nem toda a vizinhança é analisada em cada passo do VND_1. Por isso, tal solução é submetida a uma busca local mais efetiva, no caso, outro VND, denotado por VND_2. Nessa busca local, a exploração do espaço de soluções é realizada de acordo com os seguintes passos:

1. Descida Primeiro de Melhora usando vizinhança N^T;
2. Descida completa usando vizinhança N^{RB};

Se no passo 1, nenhum dos vizinhos da vizinhança N^T representar uma melhora na solução global, passa-se para o passo 2.

No passo 2, inicialmente testam-se todas as possíveis realocações de blocos com apenas uma tarefa e, quando não for mais possível melhorar a solução com um determinado tamanho de bloco, passa-se a explorar movimentos com blocos de tamanho imediatamente superior, até que o tamanho dos blocos atinja seu valor máximo, dado por $n-1$. Se uma solução de melhora é encontrada, volta-se ao passo 1. O procedimento é interrompido quando um ótimo local com relação às vizinhanças N^T e N^{RB} é encontrado. Observa-se que a vizinhança N^R corresponde a realocar blocos com apenas uma tarefa. Logo, a vizinhança N^R está contida na vizinhança N^{RB} e um ótimo local em relação à esta última também é um ótimo local em relação à primeira.

3.8 Redução do Espaço de Busca - GPV-REB

A avaliação de todos os vizinhos gerados em uma busca local requer um grande e muitas vezes injustificável esforço computacional (França Filho, 2007). Por isso, é interessante utilizar estratégias que tenham por objetivo evitar a avaliação de soluções que apresentam determinadas características julgadas ruins e, dessa forma, reduzir o esforço computacional. Como consequência, pode-se reduzir o tempo computacional para resolver o problema ou, fixado um dado tempo de processamento, mais soluções promissoras podem ser exploradas. Esta estratégia permite, assim, resolver problemas de dimensões maiores.

Se duas tarefas $i, j \in I$ satisfazem as condições (2), (3), (4) e (5), espera-se que boas soluções possuam a tarefa i sequenciada antes da tarefa j.

$$E_i \leq E_j \quad (2)$$
$$\alpha_i \leq \alpha_j \quad (3)$$
$$T_i \leq T_j \quad (4)$$
$$\beta_i \geq \beta_j \quad (5)$$

Portanto, se i é sequenciada antes de j na sequência corrente, é proposta a seguinte estratégia de redução do espaço de busca:

- Durante a exploração da vizinhança N^T, a solução produzida pela troca das ordens de processamento da tarefa i com a tarefa j é rejeitada sem ser avaliada
- Durante a exploração das vizinhanças N^R ou N^{RB}, as realocações que levam a tarefa i a ser sequenciada depois da tarefa j são descartadas sem serem avaliadas

A Figura 3 ilustra a situação em que $E_3 \leq E_2$, $\alpha_3 \leq \alpha_2$, $T_3 \leq T_2$ e $\beta_3 \geq \beta_2$. Nesse caso, as soluções obtidas pela troca de ordem de processamento da tarefa 3 com a tarefa 2 ou pelas realocações, de tarefas ou de blocos, que levam a tarefa 3 a ser sequenciada depois da tarefa 2, não são avaliadas por não serem consideradas soluções promissoras.

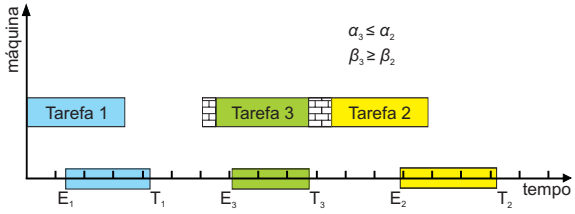

Figura 3: Exemplo de situação em que a estratégia de redução do espaço de busca é utilizada.

Essa estratégia de redução do espaço de busca foi incorporada ao algoritmo GPV, e o algoritmo resultante foi denotado por GPV-REB. Ele se diferencia do GPV unicamente na aplicação das buscas locais da primeira fase deste, as quais são feitas pelo método de descida randômica usando vizinhança N^R (usado para refinamento das soluções parciais), bem como pelo VND_1 (usado para refinamento da solução construída). A segunda fase dos dois algoritmos são iguais.

4 Resultados computacionais

Os algoritmos heurísticos propostos, GPV e GPV-REB, foram implementados em linguagem C, usando-se o compilador Borland C++ Builder 5.0. Para testá-los, foi utilizado o conjunto de problemas-teste BDNS, disponível em http://www.iceb.ufop.br/decom/prof/marcone/projects/scheduling/instances.htm. Este conjunto envolve grupos de problemas-teste com 6, 7, 8, 9, 10, 11, 12, 15, 20, 30, 40, 50, 75, 100 e 150 tarefas e foram gerados de acordo com a metodologia apresentada nos trabalhos de Wan and Yen (2002) e Rabadi et al. (2004).

Nesse conjunto, dada uma tarefa $i \in I$, o tempo de processamento (P_i), o custo por unidade de atraso (β_i) e o custo por unidade de antecipação (α_i), são valores inteiros selecionados, aleatoriamente e com distribuição uniforme, dentro dos intervalos $[1, 100]$, $[1, 10]$ e $[1, \beta_i]$, respectivamente. O centro da janela de entrega de i é um valor inteiro aleatório no intervalo $\left[\left(1 - FA - \frac{VRJ}{2}\right) \times TTP, \left(1 - FA + \frac{VRJ}{2}\right) \times TTP\right]$, sendo TTP o tempo total de processamento de todas as tarefas, FA o fator de atraso e VRJ a variação relativa da janela de entrega. São utilizados os valores 0,1; 0,3; 0,5 e 0,8 para FA e 0,4; 0,7; 1,0 e 1,3 para VRJ. O tamanho da janela de entrega de i é um número inteiro selecionado, aleatoriamente e com distribuição uniforme, no intervalo $\left[0, \frac{TTP}{n}\right]$, em que n é o número de tarefas de I. Para toda tarefa $j \in I$ e $j \neq i$, o tempo de preparação (S_{ij}) é um inteiro aleatório pertencente ao intervalo $[5, 30]$. Como são utilizados 4 valores distintos de TTP e 4 de VRJ, são 16 problemas-teste de cada tamanho, totalizando 240 problemas-teste.

Para auxiliar na apresentação e comparação dos resultados, foram utilizadas duas medidas de desempenho, dadas pelas equações (6) e (7). Nessas equações, para um dado problema-teste i, f_i^* é o valor da função objetivo da melhor solução conhecida para o problema e $f_i^{X^\star}$ e $\overline{f}_i^X$ são, respectivamente, o valor da função objetivo da melhor solução encontrada e a média dos valores da função objetivo aplicada nas soluções encontradas durante todas as execuções do algoritmo X para o problema-teste i.

$$dev_i^{best} = \frac{f_i^{X^\star} - f_i^\star}{f_i^\star} \quad (6)$$

$$dev_i^{avg} = \frac{\overline{f}_i^X - f_i^\star}{f_i^\star} \quad (7)$$

A primeira equação calcula o desvio do valor da função objetivo da melhor solução encontrada pelo algoritmo X em relação ao valor da função objetivo da melhor solução conhecida para o respectivo problema-teste. Já a segunda equação calcula a variabilidade média dos valores da função objetivo das soluções encontradas pelo algoritmo X em relação ao valor da melhor solução conhecida para o respectivo problema.

Os parâmetros utilizados estão resumidos na Tabela 1, na qual n representa o número de tarefas a serem sequenciadas.

Tabela 1: Parâmetros utilizados.

Parâmetros	Valores
γ	0,12
$MDRmax$	$7 \times n$, se $n \leq 50$ e $5 \times n$, se $n > 50$
$VNDmax$	$11 \times n$, se $n \leq 50$ e $5 \times n$, se $n > 50$
$GRASPmax$	24, se $n \leq 50$ e 12, se $n > 50$

Cada problema foi resolvido 30 vezes, exceto aqueles com mais de 50 tarefas, os quais foram resolvidos apenas 10 vezes devido ao maior tempo computacional demandado. Os resultados obtidos pelos algoritmos GPV e GPV-REB foram comparados apenas com os do Algoritmo Genético de

Ribeiro (2009), pois somente esse autor realizou experimentos com essa base de dados.

Uma síntese dos resultados encontrados pelos algoritmos GPV e GPV-REB e pelo algoritmo de Ribeiro (2009) é apresentada na Tabela 2. Nela, a primeira coluna indica o número de tarefas dos problemas de cada grupo de problemas-teste. Nas colunas "Ribeiro (2009)", "GPV" e "GPV-REB" são apresentados os resultados obtidos pelos respectivos algoritmos. Para um grupo de problemas-teste com mesmo número de tarefas, as colunas "Tempo" mostram as médias dos tempos demandados (em segundos) pelos respectivos algoritmos, as colunas "$\overline{dev}^{best}$" apresentam as médias dos dev_i^{best}'s (em porcentagem) obtidos pelos respectivos algoritmos e as colunas "$\overline{dev}^{avg}$" apresentam as médias dos dev_i^{avg}'s obtidos pelos respectivos algoritmos. Os resultados de Ribeiro (2009) relativos aos problemas-teste com 150 tarefas não são apresentados porque tal autor não resolveu tais problemas.

De acordo com a Tabela 2, o GPV obteve valores de $\overline{dev}^{best}$ sempre menores ou iguais aos do algoritmo de Ribeiro (2009). Isso significa que as soluções produzidas pelo GPV foram, em média, melhores ou iguais às produzidas pelo algoritmo de tal autor. Ambos os algoritmos encontraram $\overline{dev}^{avg}$'s próximos de zero nos conjuntos de problemas-teste com até 20 tarefas. Já nos conjuntos de problemas com mais de 20 tarefas, os valores dos $\overline{dev}^{avg}$'s obtidos pelo GPV foram sempre menores que os do algoritmo de Ribeiro (2009). Em relação aos tempos de processamento, uma comparação justa não é possível, uma vez que os algoritmos foram testados em máquinas diferentes. No entanto, o algoritmo de Ribeiro (2009) demandou um tempo significativamente maior que os requeridos pelo GPV para todos os conjuntos de problemas-teste. Para os problemas envolvendo 100 tarefas, por exemplo, o tempo médio demandado pelo Algoritmo Genético de Ribeiro (2009) foi aproximadamente 4,8 vezes maior que aquele requerido pelo GPV.

Na Tabela 2 também é possível observar que o algoritmo GPV obteve valores de $\overline{dev}^{best}$ menores que 1% em todos os conjuntos de problemas-teste, exceto para o de 75 tarefas. Mesmo assim, para este último conjunto, o $\overline{dev}^{best}$ foi menor que aquele obtido pelo algoritmo de Ribeiro (2009). Os $\overline{dev}^{avg}$'s foram relativamente baixos, sendo o maior valor igual a 6,66% e, para os problemas com 150 tarefas, próximo do valor obtido pelo GPV. Os tempos médios demandados também foram menores que os demandados pelos outros dois algoritmos. Se comparado aos tempos médios requeridos pelo GPV, os demandados pelo GPV-REB foram de 34% a 50% menores.

Finalmente, destaca-se que os tempos computacionais – mesmo os do algoritmo GPV-REB – ainda são elevados quando comparados com aqueles encontrados na literatura para resolver problemas de sequenciamento de tarefas. O motivo está no fato de que o PSUMAA-JT é uma generalização dos problemas de sequenciamento e nele há a necessidade de se determinar as datas ótimas de início de processamento para cada tarefa de uma sequência gerada, o que eleva consideravelmente o tempo do algoritmo.

5 Conclusões

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

Propôs-se um algoritmo heurístico para determinar a melhor sequência de produção. Tal algoritmo, denominado GPV, é composto de duas fases. Na primeira, constrói-se uma solução com base no procedimento de construção da metaheurística GRASP e no Princípio da Otimalidade Próxima (POP), sendo esta solução posteriormente refinada por uma Descida em Vizinhança Variável (*Variable Neighborhood Descent* - VND). Na segunda fase, faz-se o pós-refinamento da solução proveniente da fase anterior por meio de outro VND. Para cada sequência gerada pelo algoritmo, um procedimento de complexidade polinomial encontrado na literatura é acionado para determinar as datas ótimas de conclusão das tarefas. Com o objetivo de reduzir o custo computacional, também foi proposta uma estratégia de redução do espaço de busca. Essa estratégia foi utilizada durante a primeira fase do GPV, originando, assim, o algoritmo denominado GPV-REB.

Os algoritmos GPV e GPV-REB foram aplicados em um conjunto genérico de problemas-teste. Os resultados encontrados por esses algoritmos foram comparados com um Algoritmo Genético Adaptativo proposto em Ribeiro (2009), que foi o único autor a utilizar também esta base de dados. O GPV mostrou-se mais eficiente que o algoritmo de tal autor, encontrando soluções de melhor qualidade e em tempo computacional muito menor. Por outro lado, os tempos computacionais demandados pelo algoritmo GPV-REB foram sempre significativamente menores que aqueles despendidos pelo GPV. Apesar de as soluções encontradas pelo GPV-REB serem um pouco inferiores àquelas encontradas pelo GPV, esta diferença diminui à medida em que se aumenta o número de tarefas do problema. Isso sugere que vale a pena investir em técnicas e propriedades que permitam reduzir o espaço de busca.

Como trabalhos futuros, sugere-se: (*i*) comparar o GPV com o GRASP tradicional a fim de evidenciar os ganhos; (*ii*) fazer um estudo compa-

Tabela 2: Comparação de resultados Ribeiro (2009) × GPV × GPV-REB no conjunto BDNS.

♯	Ribeiro (2009)			GPV			GPV-REB		
Tarefas	$\overline{dev}^{best}$ (%)	$\overline{dev}^{avg}$ (%)	**Tempo**[1] (s)	$\overline{dev}^{best}$ (%)	$\overline{dev}^{avg}$ (%)	**Tempo**[2] (s)	$\overline{dev}^{best}$ (%)	$\overline{dev}^{avg}$ (%)	**Tempo**[2] (s)
006	0,00	0,00	0,76	0,00	0,00	0,04	0,00	0,00	0,03
007	0,00	0,00	0,76	0,00	0,00	0,07	0,00	0,00	0,03
008	0,00	0,00	1,03	0,00	0,00	0,30	0,00	0,76	0,25
009	0,00	0,00	1,35	0,00	0,00	0,37	0,01	0,18	0,29
010	0,00	0,00	1,61	0,00	0,20	0,45	0,47	0,86	0,34
011	0,00	0,00	2,20	0,00	0,01	0,40	0,11	0,16	0,40
012	0,00	0,00	3,32	0,00	0,00	0,52	0,00	0,43	0,42
015	0,00	0,01	7,49	0,00	0,01	1,26	0,00	0,54	0,73
020	0,00	0,30	23,88	0,00	0,10	4,52	0,49	1,83	2,85
030	0,19	1,40	172,51	0,01	0,58	36,06	0,51	1,99	18,17
040	0,18	1,17	801,67	0,06	0,79	132,94	0,36	1,51	67,04
050	0,30	1,45	1575,11	0,06	1,28	440,49	0,65	2,57	215,60
075	5,90	5,90	4978,02	3,10	5,64	1187,76	4,14	6,66	636,86
100	2,36	2,62	22299,79	0,21	2,32	4612,79	0,57	3,79	3015,16
150	-	-	-	0,14	2,53	49524,76	0,83	2,62	32833,61

[1]Testes realizados em um PC Pentium Core 2 Duo 2,1 GHz, com 4 GB de memória RAM.
[2]Testes realizados em um PC Intel Core 2 Duo (E8400) 2.99 GHz, com 2 GB de memória RAM.

rativo do GPV com e sem o VND_2 como estratégia de pós-otimização; (*iii*) testar a aplicação do VND_2 como busca local do GRASP e não como estratégia de pós-otimização; (*iv*) testar, também, tal procedimento juntamente com a estratégia de redução do espaço de busca; (*v*) testar novas estratégias de redução do espaço de busca e (*vi*) fazer um estudo de qual tamanho de bloco proporciona melhores resultados durante a exploração da vizinhança N^{RB}.

Agradecimentos

Os autores agradecem à FAPEMIG e ao CEFET-MG pelo apoio a esta pesquisa.

Referências

Allahverdi, A., Gupta, J. N. D. and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations, *Omega: The 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.

França Filho, M. F. (2007). *GRASP e Busca Tabu aplicados a problemas de programação de tarefas em máquinas paralelas*, Tese de doutorado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Campinas.

Glover, F. and Laguna, M. (1997). *Tabu Search*, Kluwer Academic Publishers, Boston.

Gomes Júnior, A. C. (2007). *Problema de sequenciamento em uma máquina com penalidades por antecipação e atraso: Modelagem e resolução*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia de Produção, UFMG, Belo Horizonte.

Koulamas, C. (1996). Single-machine scheduling with time windows and earliness/tardiness penalties, *European Journal of Operational Research* **91**: 190–202.

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

Penna, P. H. V. (2009). *Um algoritmo heurístico híbrido para minimizar os custos com a antecipação e o atraso da produção em ambientes com janelas de entrega e tempos de preparação dependentes da sequência*, Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Mineral, UFOP, Ouro Preto.

Rabadi, G., Mollaghasemi, M. and Anagnostopoulos, G. C. (2004). A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time, *Computers & Operations Research* **31**: 1727–1751.

Ribeiro, F. F. (2009). *Um algoritmo genético adaptativo para a resolução do problema de sequenciamento em uma máquina com penalização por antecipação e atraso da produção*, Dissertação de mestrado, Programa de Pós-Graduação em Modelagem Matemática e Computacional, CEFET-MG, Belo Horizonte.

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.