UMA NOVA FORMULAÇÃO DE PROGRAMAÇÃO MATEMÁTICA INDEXADA NO TEMPO PARA UMA CLASSE DE PROBLEMAS DE SEQUENCIAMENTO EM UMA MÁQUINA

Bruno Ferreira Rosa
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Programa de Pós-Graduação em Modelagem Matemática e Computacional
Av. Amazonas, 7675 - CEP 30510-000, Belo Horizonte - MG
brunofazmat@dppg.cefetmg.br

Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto (UFOP)
ICEB, Programa de Pós-Graduação em Engenharia Mineral
CEP 35400-000, Ouro Preto - MG
marcone@iceb.ufop.br

RESUMO

Este trabalho trata do problema de sequenciamento de tarefas em uma máquina. Considera-se o tempo de preparação da máquina dependente da sequência de produção e que cada tarefa está associada a um tempo de processamento e 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 um modelo de programação linear inteira mista indexado no tempo para representar o problema. A estimativa do horizonte de planejamento, a qual faz parte dos dados de entrada do modelo, é obtida pela aplicação de um algoritmo heurístico baseado nos procedimentos GRASP, Princípio da Otimalidade Próxima e Descida em Vizinhança Variável. Experimentos computacionais mostram que o modelo utilizado, associado ao algoritmo heurístico proposto, permite resolver problemas maiores e com maior eficiência, quando comparado com outra formulação de programação matemática da literatura.

PALAVRAS CHAVE. Sequenciamento em uma máquina. Programação linear inteira mista. Formulação indexada no tempo. Otimização Combinatória.

ABSTRACT

This work deals with the problem of job scheduling in a single-machine. In the problem considered, the machine's setup time is sequence-dependent and each job has a processing time and a due window in which it should preferably be completed. The objective is to minimize the weighted sum of the tardiness and earliness in the execution of such jobs. A time-indexed mixed integer linear programming model is proposed to represent the problem. The estimate of the planning horizon, which is part of the model's inputs, it is obtained by an algorithm based on GRASP, Proximate Optimality Principle and Variable Neighborhood Descent. Computational experiments show that the new formulation, associated with the proposed algorithm, solves problems of larger dimension and with better efficiency, when compared with a model of the literature.

KEYWORDS. Single-machine sequencing. Mixed Integer Linear Programming. Time-indexed Formulation. Combinatorial Optimization.

1. Introdução

O surgimento do sistema de administração *Just in Time* (JIT), que ocorreu nos meados da década de 70, evidenciou a importância de um planejamento criterioso das atividades produtivas. Visando a redução dos custos provenientes de processos produtivos, a filosofia JIT desencoraja, além dos atrasos, também as antecipações das tarefas. Há antecipação de uma tarefa quando ela é concluída antes da data desejada para sua entrega e há atraso quando ela é concluída após tal data. Deste modo, ambas as situações acarretam penalidades.

Segundo Liaw (1999), concluir uma tarefa com atraso pode resultar em multas contratuais, perda de credibilidade da empresa ou redução de vendas. Do mesmo modo, 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 (FRANÇA FILHO, 2007).

O problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção, de agora em diante denotado por 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. Tal problema, segundo Baker e Scudder (1990), reflete melhor ambientes de produção administrados de acordo com a filosofia JIT.

Com relação às datas de entrega das tarefas, o PSUMAA pode ser dividido em três variantes:

i) datas de entrega comuns (*common due date*): todas as tarefas devem ser preferencialmente concluídas em uma única data pré-determinada;

ii) datas de entrega distintas (*distinct due dates*): existe uma data de entrega específica associada a cada tarefa, na qual a tarefa deve ser preferencialmente concluída;

iii) janelas de entrega distintas (*distinct due windows*): há um determinado período de tempo associado a cada tarefa, dentro do qual a tarefa deve ser preferencialmente concluída.

O caso de janelas de entrega distintas ocorre quando existem tolerâncias em torno das datas desejadas para a entrega de cada tarefa (KOULAMAS, 1996). Estas tolerâncias estão relacionadas às características individuais das tarefas e influenciam nos tamanhos das janelas de entrega. As tarefas concluídas dentro de suas respectivas janelas de entrega não incorrem nenhum custo. Já aquelas concluídas fora de suas janelas de entrega, são penalizadas. A produção de bens perecíveis é um exemplo relacionado. Assuma que um fabricante de produtos químicos combina certa substância A, que deteriora rapidamente, com uma segunda substância B para produzir um produto C. Se A for produzida suficientemente antes de B, ela se deteriorará. Por outro lado, se A for produzida muito depois de B, o custo da produção de C será maior.

Nas indústrias em que são produzidos diferentes tipos de produtos e existe uma troca frequente do tipo de tarefa executada em uma máquina, após a conclusão de uma tarefa, geralmente é necessário preparar a máquina antes do início da execução da tarefa seguinte. Este tempo de preparação, chamado tempo de *setup*, inclui os tempos gastos para trocar as ferramentas, preparar o material, limpar a máquina, etc. A maioria dos trabalhos em problemas de sequenciamento assume que os tempos de *setup* são independentes da sequência de produção, isto é, que eles são desprezíveis ou podem ser acrescentados aos tempos de processamento das tarefas (GUPTA e SMITH, 2006). No entanto, de acordo com Panwalkar *et al.* (1973), *apud* Gupta e Smith (2006), em grande parte das situações práticas, tais tempos são dependentes da sequência de produção. Christofoletti (2002) cita o exemplo de uma fábrica de papel que produz diversos tipos de folhas com diferentes cores, espessuras e texturas e que realiza frequentemente a preparação das máquinas para obter os diferentes tipos de produtos. Em problemas como este, os tempos de *setup* variam de acordo com sequência de produção e representam uma parcela de tempo considerável em relação ao tempo total de processamento. Portanto, eles não podem ser desconsiderados.

Considerações sobre a continuidade do funcionamento da máquina também podem ser impostas ao PSUMAA. Parte dos trabalhos da literatura não permite a inserção de tempos ociosos na sequência de produção (CHANG, 1999). Conforme Li (1997), existem casos em que o custo

por manter a máquina inativa é maior que o preço pago pela antecipação de uma tarefa e, neste caso, vale a pena antecipar a produção. Outra situação em que isto ocorre é quando a capacidade da máquina é inferior à demanda. No entanto, há casos em que vale a pena manter a máquina parada, mesmo que exista uma tarefa disponível para ser processada (SOUZA *et al.*, 2008b).

O PSUMAA possui muitas aplicações práticas em indústrias metalúrgicas, têxteis, de tintas, entre outras. O caso real de uma indústria siderúrgica é tratado por Bustamante (2006). Em tal trabalho, uma máquina é considerada como sendo uma sequência de laminadores e cada tarefa representa a produção de um determinado produto (barra chata, cantoneira, vergalhão etc.). Antes da fabricação de cada produto é necessário realizar um conjunto de ajustes de mesma natureza na sequência de laminadores. Estes ajustes dependem do produto a ser fabricado e do produto fabricado anteriormente. Como cada ajuste exige um tempo de execução, a soma destes tempos configura o tempo de *setup*.

Este trabalho trata o PSUMAA com janelas de entrega distintas, tempo de preparação dependente da sequência de produção e são permitidos tempos ociosos entre as execuções de tarefas consecutivas. Para evitar dúvidas, o PSUMAA com tais características será doravante denotado por PSUMAA-JE-TP. Dentre os trabalhos que tratam este problema, a maioria tem seu foco em procedimentos heurísticos para resolvê-lo. Apenas nos trabalhos de Bustamante (2006) e Gomes Jr. *et al*. (2007) são propostos métodos exatos para resolvê-lo.

Neste trabalho, um modelo de programação linear inteira mista indexado no tempo é utilizado para representar o PSUMAA-JE-TP. Para mensurar o horizonte de planejamento de tal modelo, propõe-se um algoritmo heurístico baseado nos métodos GRASP (FEO e RESENDE, 1995), Princípio da Otimalidade Próxima - POP (GLOVER e LAGUNA, 1997) e *Variable Neighborhood Descent* - VND (MLADENOVIC e HANSEN, 1997). O referido modelo é resolvido pelo otimizador CPLEX e comparado com outra formulação da literatura.

O restante deste trabalho está organizado como segue. Na Seção 2 são apresentados trabalhos relacionados ao PSUMAA-JE-TP. Na Seção 3 faz-se uma descrição detalhada do problema tratado. A formulação de programação matemática indexada no tempo é descrita na Seção 4 e na seguinte detalha-se o algoritmo proposto para determinar o horizonte de planejamento de tal formulação. Na Seção 6 são apresentados e discutidos os resultados computacionais, enquanto a Seção 7 conclui o trabalho.

2. Trabalhos Relacionados

Além de possuir um grande número de aplicações práticas, problemas de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção são difíceis de ser resolvidos na otimalidade (ALLAHVERDI *et al.*, 1999). Tal dificuldade cresce explosivamente na medida em que se aumenta a quantidade de tarefas a serem sequenciadas. Esta união entre aplicabilidade e dificuldade de resolução motiva a pesquisa de algoritmos eficientes para a resolução desta classe de problemas. Deste modo, muitos trabalhos são encontrados na literatura com o objetivo de resolver casos particulares do PSUMAA-JE-TP tratado neste trabalho. Uma breve descrição de trabalhos correlatos é apresentada a seguir.

Alidaee e Dragan (1997) tratam do caso com datas de entrega comuns. Estes autores consideram as penalidades por antecipação e por atraso da produção proporcional ao tempo de processamento da tarefa e iguais para a mesma tarefa. Os autores propõem um algoritmo de complexidade $O(n \log n)$ que resolve este problema na otimalidade.

Ying (2008) e Rabadi *et al*. (2004) também trataram do PSUMAA com datas de entrega comuns. O primeiro autor resolve o problema com tempos de *setup* independentes da sequência de produção por meio de um algoritmo *branch-and-bound* que faz uso de um procedimento *Recovering Beam Search* para caminhar na árvore de busca. No segundo trabalho é proposto um algoritmo *branch-and-bound* para obter soluções do caso com tempos de *setup* dependentes da sequência de produção e com soma dos tempos de processamento das tarefas não maior que a data de entrega. Os autores resolveram problemas-teste com até 25 tarefas em tempo

computacional aceitável, o que representou um avanço para a época, já que os algoritmos exatos de então resolviam, na otimalidade, apenas problemas desta classe com até 8 tarefas.

Li (1997) estudou o PSUMAA com datas de entrega distintas, tempos de setup independentes da sequência de produção e sem a permissão de tempos ociosos na máquina. O problema é decomposto em dois subproblemas com estruturas mais simples. O limite inferior do problema é, então, dado pela soma dos limites inferiores dos dois subproblemas, cada qual obtido por relaxação lagrangeana. Também são desenvolvidos dois procedimentos de ajuste de multiplicadores, com complexidade $O(n \log n)$, para resolver os duais dos subproblemas. Um algoritmo *branch-and-bound* baseado nesses procedimentos é apresentado e utilizado para resolver problemas-teste com até 50 tarefas, dobrando a dimensão dos problemas resolvidos na otimalidade por algoritmos exatos até aquela data. O autor também propôs uma heurística baseada em busca local para resolver problemas-teste com até 3000 tarefas. Valente e Alves (2005) propuseram duas novas heurísticas, sendo uma regra de despacho e um procedimento guloso, para o mesmo problema. Também são utilizadas regras de dominância, visto que elas melhoram os resultados das heurísticas e demandam pouco tempo adicional.

Lee e Choi (1995) abordaram o PSUMAA com datas de entrega distintas, tempos de *setup* independentes da sequência de produção e com permissão de tempos ociosos. É apresentado um algoritmo de complexidade polinomial para determinar a data ótima de início de uma tarefa em uma dada sequência de produção. Um Algoritmo Genético que faz uso deste algoritmo de data ótima também é apresentado. O mesmo problema é estudado por Mazzini e Armentano (2001). Uma heurística construtiva que determina a sequência de produção e, simultaneamente, insere tempos ociosos é proposta por estes autores.

No trabalho de Bustamante (2006) é estudado o PSUMAA com tempos de *setup* dependentes da sequência de produção, datas de entrega distintas e é permitida a ociosidade de máquina. São desenvolvidos dois modelos de programação linear inteira mista, sendo que tais modelos exigem que os tempos de *setup* satisfaçam a desigualdade triangular. O autor também sugere alterações nos modelos que permitem resolver o caso com janelas de entrega distintas, mas os testes computacionais foram realizados apenas em problemas-teste com datas de entrega distintas. Foram resolvidos na otimalidade problemas com até 10 tarefas, por meio do *software* de otimização GLPK, versão 4.8.

Wan e Yen (2002) trataram o PSUMAA com janelas de entrega distintas, tempos de *setup* independentes da sequência de produção, permitindo tempos ociosos de máquina. Primeiramente, foi apresentada uma formulação matemática do problema junto com várias propriedades importantes para sua resolução. Em seguida, foi proposto um procedimento de complexidade polinomial para determinar a data de conclusão ótima de processamento de cada tarefa em uma dada sequência, sendo este procedimento uma extensão do algoritmo proposto por Lee e Choi (1995). Por fim, uma Busca Tabu (*Tabu Search*, TS), que faz uso do procedimento de datas ótimas, é proposta para resolver o problema. Foram realizados testes em problemas envolvendo até 80 tarefas.

Koulamas (1996) focou o problema com janelas de entrega distintas, porém com uma sensível distinção dos demais trabalhos da literatura em relação à antecipação. Em seu trabalho, há antecipação de uma tarefa quando seu processamento é iniciado antes do início de sua janela de entrega. O restante da literatura considera que há antecipação quando uma tarefa é concluída antes do início de tal janela. O autor também considerou tempos de *setup* independentes da sequência de produção, sendo permitida a ociosidade de máquina. O autor adaptou heurísticas já utilizadas em outros casos do PSUMAA e as aplicou nessa versão. Também foi proposto um algoritmo que insere tempos ociosos, de modo ótimo, em uma dada sequência de produção. Foram realizados testes em problemas com até 200 tarefas.

Gomes Jr. *et al*. (2007), Souza *et al*. (2008a) e Souza *et al*. (2008b) foram os únicos trabalhos encontrados para resolver o PSUMAA-JE-TP. No primeiro trabalho é proposto um modelo de programação matemática para representar o problema, no qual deixa de ser necessário que o problema satisfaça a desigualdade triangular. Também é proposto um método heurístico baseado em GRASP, *Iterated Local Search* 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 que determina a data ótima de início de processamento das tarefas na sequência dada, sendo este algoritmo uma extensão daquele proposto por Wan e Yen (2002). São realizados experimentos computacionais em problemas-teste com os tempos de *setup* simétricos e com até 75 tarefas, utilizando a heurística, e até 12 tarefas, utilizando o modelo matemático. O segundo trabalho apresenta uma heurística baseada em GRASP, VND, TS e Reconexão por Caminhos (*Path Relinking*, PR); enquanto no terceiro são apresentados os resultados detalhados do algoritmo proposto. Nos dois últimos trabalhos são realizados experimentos nos mesmos problemas-teste utilizados por Gomes Jr. *et al*. (2007), sendo mostrada a superioridade do algoritmo proposto em relação a este último.

3. Descrição do Problema

O PSUMAA-JE-TP abordado neste trabalho possui as seguintes características:

i) Uma máquina deve processar um conjunto I de n de tarefas;

ii) Associado a cada tarefa $i \in I$ está:

- *a)* Um tempo de processamento P_i;
- *b)* Uma janela de entrega $[E_i, T_i]$, dentro da qual esta tarefa deve ser preferencialmente concluída;
- *c)* Um custo α_i por unidade de tempo de antecipação;
- *d)* Um custo β_i por unidade de tempo de atraso;

iii) Há antecipação de uma tarefa $i \in I$ quando seu processamento é concluído antes de E_i;

iv) Há atraso de uma tarefa $i \in I$ quando seu processamento é concluído depois de T_i;

v) As tarefas que forem concluídas dentro de suas respectivas janelas de entrega não geram penalidades;

vi) A máquina executa no máximo uma tarefa por vez e, uma vez iniciado o processamento de uma tarefa, não é permitida a sua interrupção;

vii) Todas as tarefas estão disponíveis para processamento na data 0;

viii) Entre duas tarefas i e $j \in I$ consecutivas, é necessário um tempo S_{ij} de preparação da máquina, chamado tempo de *setup*. Assume-se que o tempo de preparação da máquina para o processamento da primeira tarefa na sequência é igual a zero;

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

O objetivo é determinar uma sequência de produção e as datas de início de produção das tarefas de sorte a minimizar a soma ponderada das antecipações e atrasos.

4. Formulação Matemática

O modelo de programação matemática utilizado para representar o PSUMAA-JE-TP é baseado na formulação indexada no tempo proposta em de Paula (2008). Este modelo, doravante denotado por MPMDT, faz uma discretização do tempo e pode ser descrito como segue.

Considere que o conjunto $H = \{h_0, h_1, h_2, ..., h_L\}$ representa o horizonte de planejamento para o processamento das tarefas e sejam x_{ih} variáveis binárias que determinam a sequência de produção, sendo

$$x_{ih} = \begin{cases} 1, & \text{se a tarefa } i \text{ é programada pra iniciar na data } h; \\ 0, & \text{caso contrário.} \end{cases}$$

para todo $i \in I$ e para todo $h \in H$.

Se e_i e t_i representam, respectivamente, as unidades de tempo de antecipação e de tempo de atraso da tarefa $i \in I$, o MPMDT pode ser formulado pelas equações (1) a (8).

$$\min \quad z = \sum_{i=1}^{n} (\alpha_i e_i + \beta_i t_i) \tag{1}$$

$$\text{s. a.} \quad x_{ih} + \sum_{u=h}^{\min(h+P_i+S_{ij}-1, H_L)} x_{ju} \leq 1, \qquad \forall i, j \in I, \ \forall h \in H \ \text{ e } \ i \neq j \tag{2}$$

$$\sum_{h=H_0}^{H_L} x_{ih} = 1, \qquad \forall i \in I \tag{3}$$

$$\sum_{h=H_0}^{H_L} h x_{ih} + P_i + e_i \geq E_i, \qquad \forall i \in I \tag{4}$$

$$\sum_{h=H_0}^{H_L} h x_{ih} + P_i - t_i \leq T_i, \qquad \forall i \in I \tag{5}$$

$$e_i \geq 0, \qquad \forall i \in I \tag{6}$$

$$t_i \geq 0, \qquad \forall i \in I \tag{7}$$

$$x_{ih} \in \{0,1\}, \qquad \forall i \in I \ \text{ e } \ \forall h \in H \tag{8}$$

A função objetivo, representada pela equação (1), tem como critério de otimização a minimização da soma ponderada das antecipações e dos atrasos. As restrições (2) garantem que existe tempo suficiente para executar uma tarefa i e preparar a máquina antes do início do processamento da tarefa seguinte j. As restrições (3) garantem que cada tarefa seja executada uma, e somente uma vez. As restrições (4) e (5) definem as antecipações e os atrasos de acordo com as respectivas janelas de entrega de cada tarefa. As restrições (6), (7) e (8) estão associadas aos domínios das variáveis do problema.

É importante observar que esta formulação somente é válida se o problema satisfizer a desigualdade triangular, ou seja, se as condições (9) a seguir forem satisfeitas:

$$S_{ik} \leq S_{ij} + S_{jk} + P_j, \qquad \forall i, j, k \in I, \ i \neq j, \ i \neq k \ \text{e} \ j \neq k \tag{9}$$

No PSUMAA-JE-TP abordado, a execução de qualquer tarefa pode ser iniciada em qualquer momento futuro. Portanto, caso se tenha a garantia de que na sequência ótima o início do processamento da primeira tarefa não ocorre antes de h_{inf} e que o início do processamento da última tarefa é no máximo h_{sup}, é conveniente adotar $H = \{h_{inf}, h_{inf+1}, h_{inf+2}, ..., h_{sup}\}$. Deste modo, como o MPMDT é fortemente sensível à cardinalidade do conjunto H, quanto menor for o intervalo $[h_{inf}, h_{sup}]$, menor será o número de variáveis do modelo.

5. Determinação do Conjunto *H*

O MPMDT é fortemente dependente do conjunto $H = [h_{inf}, h_{sup}]$, visto que ele representa o horizonte de planejamento e, consequentemente, determina o número de variáveis e restrições deste modelo. Sendo assim, é de fundamental importância escolher tal conjunto de modo a facilitar a obtenção da solução ótima. Para tanto, a amplitude desse intervalo não deve ser muito grande, para que não se tenha muitas restrições, variáveis e um modelo mais difícil de ser resolvido na otimalidade.

Para determinar os limites do conjunto H da formulação MPMDT, desenvolveu-se o algoritmo heurístico GPV, descrito pelo pseudocódigo da Figura 1.

Este algoritmo combina os procedimentos GRASP (FEO e RESENDE, 1995), Princípio da Otimalidade Próxima - POP (GLOVER e LAGUNA, 1997) e VND (MLADENOVIC e HANSEN, 1997), e é composto de duas fases. Na primeira (linhas 1 a 12 da Figura 1), gera-se uma solução com base na metaheurística GRASP (*GRASPMax* é um parâmetro do método). Na segunda (linha 13), faz-se o pós-refinamento da solução proveniente da fase anterior. Os valores de h_{inf} e de h_{sup} (linha 14) são dados, respectivamente, pela data de início da primeira tarefa e pela

data de conclusão da última tarefa na sequência v^*. O detalhamento deste algoritmo é apresentado nas subseções seguintes.

```
Algoritmo GPV()
1    f* ← +∞;
2    Iter ← 0;
3    enquanto (Iter < GRASPMax) faça
4          Iter ← Iter + 1;
5          v0 ← ConstruaSolucao();
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*, hinf, hsup;
Fim GPV;
```

Figura 1: Algoritmo GPV

5.1. Representação de uma Solução

Uma solução (sequência) para o PSUMAA-JE-TP com n tarefas é representada por um vetor v de n posições, onde cada posição $i = 1, 2,..., n$ indica a ordem de produção da tarefa v_i. Por exemplo, dada a sequência $v = \{4, 6, 1, 5, 3, 2\}$, para o PSUMAA-JE-TP com 6 tarefas, a tarefa 4 é a primeira a ser realizada e a tarefa 2, a última.

5.2. 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 com duas ou mais tarefas. Esses movimentos definem, respectivamente, as vizinhanças N^T, N^R e N^{RB}. Sendo assim, para uma solução do PSUMAA-JE-TP com n tarefas, há $n(n\text{-}1)/2$ vizinhos na vizinhança N^T, $(n\text{-}1)^2$ vizinhos na vizinhança N^R e $(n\text{-}1)(n\text{-}2)(n\text{-}3)/6$ vizinhos na vizinhança N^{RB}.

5.3. Função de Avaliação

Como os movimentos utilizados não geram soluções inviáveis, uma sequência é avaliada pela própria função objetivo, dada pela expressão (1) do MPMDT. Para determinar os valores de e_i e t_i é utilizado o algoritmo de determinação das datas ótimas de início de processamento (ADDOIP) proposto por Gomes Jr. *et al.* (2007).

5.4. Construção de uma Solução

Nesta etapa da primeira fase do algoritmo (linha 5 da Figura 1), uma solução é formada, tarefa por tarefa, de forma parcialmente gulosa, seguindo as ideias da fase de construção do algoritmo GRASP. A cada iteração, as tarefas que ainda estão fora da solução são avaliadas por uma função g, que estima o benefício associado à sua inclusão na solução parcial. As tarefas j que possuírem $g(j)$ menor ou igual a $g_{min} + \gamma \times (g_{max} - g_{min})$, com $\gamma \in [0, 1]$, são inseridas em uma Lista Restrita de Candidatos (LRC). Desta lista é escolhida aleatoriamente uma, a qual é adicionada à solução parcial.

Para estimar o benefício da inserção de cada tarefa j ainda não sequenciada, em cada iteração i, utiliza-se uma das quatro seguintes funções como critério de seleção:

(1) $g_1(t) = 2E_j / \max\{E_k \mid k \in C\} + 2T_j / \max\{T_k \mid k \in C\} + \alpha_j / \max\{\alpha_k \mid k \in C\} - \beta_j / \max\{\beta_k \mid k \in C\} + P_j / \max\{P_k \mid k \in C\} + S_{i-1,j}/\max\{S_{i-1,k} \mid k \in C\}$, onde C representa o conjunto das tarefas ainda não sequenciadas até a *i*-ésima iteração (para *i* = 1, é utilizado $-\text{média}\{S_{kj} \mid k \in I \text{ e } k \neq j\} / \max\{S_{kl} \mid k,l \in I \text{ e } k \neq l\}$ no lugar de $S_{i-1,j}/\max\{S_{i-1,k} \mid k \in C\}$);

(2) $g_2(j) = T_j$;

(3) $g_3(j) = (\alpha_j E_j + \beta_j T_j)/(\alpha_j + \beta_j)$;

(4) $g_4(j) = E_j$.

As funções g são utilizadas conforme a ordem anterior. Nas quatro primeiras iterações, ou seja, na primeira vez em que cada função g é utilizada, faz-se $\gamma = 0$. Nas demais iterações, γ é selecionado aleatoriamente dentro de um conjunto Γ, sendo Γ um parâmetro.

Ainda na etapa de construção, é aplicado o Princípio da Otimalidade Próxima (POP). Este princípio é baseado na idéia de que boas soluções em um nível estão próximas de boas soluções em um nível adjacente (GLOVER e 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 em relação à 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 com um valor menor para a função objetivo, a nova sequência é aceita e passa a ser a solução corrente; caso contrário, é testada outra realocação. A busca é interrompida após *MRDMax* realocações consecutivas sem melhora na função objetivo, sendo *MRDMax* um parâmetro do procedimento.

O pseudocódigo da fase de construção de uma solução é apresentado na Figura 2.

```
Procedimento ConstruaSolucao();
1   v ← Ø;
2   Inicialize o conjunto C de tarefas candidatas;
3   enquanto C ≠ Ø 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       Selecione, aleatoriamente, uma tarefa t ∈ LRC;
8       v ← v ∪ {t};
9       Atualize o conjunto C de tarefas candidatas;
10      v ← MRD(v, N^R(.), MRDMax);
11  fim-enquanto;
12  Retorne v;
Fim ConstruaSolucao;
```

Figura 2: Procedimento para Construir uma Solução

5.5. VND_1

Para refinar as soluções geradas na primeira fase do algoritmo (linha 6 da Figura 1), utiliza-se a Busca em Vizinhança Variável (*Variable Neighborhood Descent* - VND). Proposto por Mladenovic e Hansen (1997), o VND é um método de busca local que consiste em explorar o espaço de busca por meio de trocas sistemáticas de estruturas de vizinhanças. Basicamente, o procedimento utilizado consiste em três passos:

(1) Método Randômico de Descida (MRD) com a vizinhança N^T;
(2) MRD com a vizinhança N^R;
(3) MRD com a vizinhança N^{RB}.

Cada passo é constituído de iterações que geram vizinhos aleatórios, com relação à respectiva estrutura de vizinhança. Sempre que uma solução de melhora é encontrada, volta-se ao primeiro passo. Quando um dos MRD's atinge *MRDmax* iterações sem melhora, passa-se para o passo seguinte. Estando-se no último passo, então o procedimento é interrompido e a melhor solução encontrada é retornada.

5.6. VND_2

Como a solução proveniente da primeira fase do algoritmo GPV (Figura 1) não é necessariamente um ótimo local em relação às vizinhanças adotadas, ela é submetida a uma busca local mais efetiva, no caso, também baseada na Busca em Vizinhança Variável (VND_2). Nesta, a exploração do espaço de soluções é realizada de acordo com os seguintes passos:

(1) Descida Completa (DC) com relação à vizinhança N^T;
(2) DC com relação à vizinhança N^R;
(3) DC na vizinhança N^{RB}.

No passo (3), em que são realizadas realocações de blocos de tarefas, inicialmente testam-se todas as realocações com blocos de duas tarefas possíveis 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 maior. Sempre que uma solução de melhora é encontrada, volta-se ao passo (1). Se em um determinado passo, todos os vizinhos com relação à respectiva estrutura de vizinhança não são de melhora, passa-se para o passo seguinte. O procedimento é interrompido quando um ótimo local com relação às três vizinhanças é encontrado.

6. Resultados Computacionais

Para testar o modelo proposto, foram gerados problemas-teste baseados nos trabalhos de Wan e Yen (2002) e Rabadi *et al*. (2004), conforme a seguir se descreve. Dada uma tarefa *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 números inteiros selecionados aleatoriamente dentro dos intervalos [1, 40], [1, 10] e [1, β_i], respectivamente. O centro da janela de entrega da tarefa *i* é um valor inteiro aleatório no intervalo [(1 - *FA* - *VRJ* / 2) × *TTP*, (1 - *FA* + *VRJ* / 2) × *TTP*], 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. O tamanho da janela de entrega é um valor inteiro selecionado aleatoriamente no intervalo [0, *TTP*/*n*], sendo *n* o número de tarefas do problema. Para toda tarefa $j \neq i$, o tempo de *setup* (S_{ij}) é um número inteiro aleatório dentro do intervalo [5, 15]. Desta forma, os tempos de *setup* não são necessariamente simétricos, como em Gomes Jr. *et al.* (2007).

Foram gerados conjuntos de problemas-teste com 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 e 16 tarefas, respectivamente, sendo 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*. Sendo assim, há 16 problemas-teste em cada conjunto.

Se um problema gerado não satisfazia a desigualdade triangular, ele era descartado e outro com os mesmos valores para *FA* e *VRJ* era gerado. Assim, todos os problemas dessa base de dados satisfazem a desigualdade triangular.

O AMPL foi utilizado para implementar as formulações MPMDT e a de Gomes Jr. *et al*. (2007). Estes modelos de programação matemática foram resolvidos pelo otimizador CPLEX, versão 10.1, da ILOG. O algoritmo para determinação do conjunto *H* utilizado no MPMDT (GPV) foi implementado na linguagem C, usando-se o compilador Dev-C++, versão 2.9.9.2. Os testes foram realizados em um computador AMD Turion(tm) 64 X2 TL-58 1900 MHz, com 2 GB de RAM, sob sistema operacional Windows XP. Os parâmetros do algoritmo GPV foram calibrados empiricamente, sendo *GRASPMax* = 16 e *MRDMax* = 7*n*. O conjunto Γ utilizado é o mesmo de Gomes Jr. *et al*. (2007), ou seja, $\Gamma = \{0,02; 0,04; 0,12; 0,14\}$.

Considerou-se, ainda, o limite de 3600 segundos para o CPLEX obter cada solução. Para os

problemas em que este limite de tempo foi atingido, a solução retornada não é necessariamente ótima; porém, é retornado, pelo CPLEX, um limite inferior para o ótimo de tal problema. Este limite foi utilizado para mensurar a qualidade da solução retornada, sendo o *gap* calculado como:

$$gap = \frac{(f^{CPLEX} - L)}{L} \times 100\% \quad (10)$$

em que f^{CPLEX} e L representam, respectivamente, o valor da solução e o limite inferior encontrados pelo otimizador. Se a solução ótima é encontrada, tem-se *gap* = 0.

Os resultados encontrados para o MPMDT e para o modelo de Gomes Jr. *et al.* (2007) são resumidos na Tabela 1. Na primeira coluna indica-se o número de tarefas do grupo de problemas-teste e na segunda, o tempo médio despendido pelo algoritmo GPV para mensurar o conjunto *H* do MPMDT. Para cada um dos modelos são apresentadas as médias dos *gaps* encontrados e as médias dos tempos demandados pelo CPLEX.

Tabela 1: Comparação modelo proposto por Gomes Jr. *et al*. (2007) × MPMDT

Número de Tarefas	**GPV**	**MPMDT**		**Gomes Jr. *et al.* (2007)**	
	Tempo Médio (s)	**Média dos *gaps* (%)**	**Tempo Médio (s)**	**Média dos *gaps* (%)**	**Tempo Médio (s)**
06	0,13	0,00	5,40	0,00	0,18
07	0,11	0,00	15,74	0,00	1,69
08	0,21	0,00	34,64	0,00	16,51
09	0,30	0,00	56,30	0,00	226,25
10	0,47	0,00	149,00	14,99	1809,11
11	0,63	0,00	412,47	41,59	2628,39
12	1,03	1,45	862,03	66,88	3378,70
13	1,53	1,77	1296,35	68,42	3075,20
14	2,23	1,56	2052,93	71,57	3267,77

Pela Tabela 1, observa-se que, por meio do modelo de programação matemática proposto por Gomes Jr. *et al*. (2007), o CPLEX somente conseguiu encontrar a solução ótima de todos os problemas-teste com até 9 tarefas. Para os problemas-teste com 10 tarefas, apesar da média dos *gaps* ser 14,99%, a solução ótima foi encontrada apenas em 62,5% dos problemas. Para os problemas-teste com mais de 10 tarefas, a média dos *gaps* foi sempre superior a 41% e a solução ótima foi encontrada em, no máximo, 31,25% dos problemas. Por outro lado, usando-se a formulação MPMDT proposta, combinada com o algoritmo GPV, o CPLEX conseguiu encontrar todas as soluções ótimas dos problemas-teste com até 11 tarefas. Além disso, o tempo computacional demandado foi menor para a resolução de problemas com 9 ou mais tarefas, mesmo adicionando-se o tempo gasto pelo algoritmo para determinar o horizonte de planejamento. Finalmente, para os casos em que a formulação proposta não foi capaz de gerar a solução ótima no tempo limite estabelecido, a média dos *gaps* foi bem menor; no caso, menor ou igual a 1,77%, contra 71,57% da formulação anterior.

Os resultados dos problemas-teste envolvendo 15 e 16 tarefas não foram apresentados porque o CPLEX aplicado à formulação MPMDT associada ao algoritmo GPV sequer foi capaz de encontrar uma solução viável em 12,5% dos problemas-teste com 15 tarefas, bem como em 31,25% dos problemas-teste com 16 tarefas. Contudo, para os casos em que essa formulação conseguiu gerar uma solução viável, os *gaps* finais médios foram bem baixos (2,66% em problemas com 15 tarefas e 4,39% nos de 16 tarefas), se comparados com aqueles produzidos pela formulação de Gomes Jr. *et al*. (2007), os quais foram sempre superiores a 70%.

7. Conclusões

Este trabalho tratou o problema de sequenciamento em uma máquina com penalidades por

antecipação e atraso da produção (PSUMAA-JE-TP), considerando janelas de entrega distintas e tempo de preparação da máquina dependente da sequência de produção. Foi utilizado um modelo de programação matemática (MPMDT), o qual faz uma discretização do tempo. Para mensurar o horizonte de planejamento *H* a ser utilizado na formulação MPMDT, foi proposto um algoritmo heurístico, denominado GPV, baseado nos procedimentos GRASP, Princípio da Otimalidade Próxima e Busca em Vizinhança Variável.

O otimizador CPLEX 10.1 foi utilizado para resolver o MPMDT e também o modelo de programação inteira mista proposto por Gomes Jr. *et al*. (2007) e foi aplicado em problemas-teste com até 16 tarefas e com tempos de *setup* não necessariamente simétricos, o que torna a base de dados mais genérica que a utilizada por Gomes Jr. *et al*. (2007). Ambos os modelos permitiram ao CPLEX encontrar a solução ótima em todos os problemas-teste com até 9 tarefas. O MPMDT associado ao algoritmo GPV se mostrou mais eficiente que o modelo de Gomes Jr. *et al*. (2007) na resolução de problemas com até 14 tarefas, visto que, para os problemas-teste com mais de 9 tarefas, o MPMDT com o algoritmo GPV proporcionou ao CPLEX encontrar soluções de melhor qualidade e em menor tempo computacional que o modelo anterior da literatura. Isto permite concluir, também, que o GPV é capaz de encontrar, em pouco tempo computacional, bons limites para o conjunto *H*, apesar de não garantir que a solução ótima está contida dentro dos limites retornados. Para os problemas com mais de 14 tarefas, os dois modelos não permitiram ao otimizador encontrar bons resultados sempre, sendo que a formulação MPMDT não foi capaz em alguns casos de encontrar uma solução viável para o problema. Apesar disso, para os problemas-teste em que uma solução viável foi encontrada, o procedimento proposto proporcionou *gaps* médios bem baixos (no máximo 1,77%), quando comparados com a formulação de Gomes Jr. *et al*. (2007), que chegou a encontrar *gap* médio de 71,57%.

Agradecimentos

O primeiro autor agradece ao CEFET-MG pela bolsa de pesquisa e o terceiro, ao CNPq (processo 474831/2007-8) e à FAPERJ (processo E-26/101.023/2007), pelo apoio ao desenvolvimento da presente pesquisa.

Referências

Alidaee, B. e Dragan, I. (1997) A note on minimizing the weighted sum of tardy and early completion penalties in a single machine: A case of small common due date, *European Journal of Operational Research*, 96, 559-563.

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

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

Bustamante, L. M., Minimização do Custo de Antecipação e Atraso para o Problema de Sequenciamento de uma Máquina com Tempo de Preparação Dependente da Sequência: Aplicação em uma Usina Siderúrgica, *Dissertação de mestrado*, Programa de Pós-Graduação em Engenharia de Produção, UFMG, Belo Horizonte, 2006.

Chang, P. C. (1999), A Branch and Bound Approach for Single Machine Scheduling with Earliness and Tardiness Penalties, *Computers & Mathematics with Applications*, 37, 133-144.

Christofoletti, L. M., Métodos de Reinício Aplicados ao Sequenciamento em Uma Máquina com Tempos de Preparação e Datas de Entrega, *Dissertação de mestrado*, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Campinas, 2002.

de Paula, M. R., Heurísticas para a Minimização dos Atrasos em Sequenciamento de Máquinas Paralelas com Tempos de Preparação Dependentes da Sequência, *Dissertação de Mestrado*, Programa de Pós-Graduação em Ciências da Computação, UFMG, Belo Horizonte, 2008.

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

França Filho, M. F., GRASP e Busca Tabu aplicados a problemas de programação e tarefas em máquinas paralelas, *Tese de doutorado*, Departamento de Engenharia de Sistemas, UNICAMP, Campinas, 2007.
Glover, F. e Laguna, M. (1997), *Tabu Search*, Kluwer Academic Publishers.
Gomes Jr., A. C., Carvalho, C. R. V., Munhoz, P. L. A. e 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, *In* Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional, SBPO, 649-1660.
Gupta, S. R. e Smith, J. S. (2006), Algorithms for single machine total tardiness scheduling with sequence dependent setups, *European Journal of Operational Research*, 175, 722-739.
Koulamas, C. (1996), Single-machine scheduling with time windows and earliness/tardiness penalties, *European Journal of Operational Research*, 91, 190-202.
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*, 22, 857-869.
Li, G. (1997), Single Machine Earliness and Tardiness Scheduling, *European Journal of Operational Research*, 96, 546–558.
Liaw, C.-F. (1999), A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem, *Computers & Operations Research*, 26, 679-693.
Mazzini, R. e Armentano, V. A. (2001), A Heuristic for Single Machine Scheduling with Early and Tardy Costs, *European Journal of Operational Research*, 128, 129-146.
Mladenovic, N. e Hansen, P. (1997), Variable Neighborhood Search, Computers and Operations Research, 24, 1097-1100.
Panwalkar, S. S., Dudek, R. A. e Smith, M. L. (1973), Sequencing research and the industrial scheduling problem, *In* Elmaghraby SE (Eds), Symposium on the Theory of Scheduling and its Applications. Springer: Berlin, 29-38.
Rabadi, G., Mollaghasemi, M. e 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.
Souza, M. J. F., Penna, P. H. V. e Gonçalves, F. A. C. A. (2008a), GRASP, VND, Busca Tabu e Reconexão por Caminhos para o problema de sequenciamento em uma máquina com tempos de preparação dependentes da sequência da produção, janelas de entrega distintas e penalidades por antecipação e atraso da produção, *In* Anais do XL Simpósio Brasileiro de Pesquisa Operacional, SBPO, 1320-1331.
Souza, M. J. F., Penna, P. H. V., Gonçalves, F. A. C. A. e Ochi, L. S. (2008b), Uma heurística híbrida para minimizar custos com antecipação e atraso em sistemas de produção com janelas de entrega e tempos de preparação dependentes da sequência, *In* Anais do XI Simpósio de Pesquisa Operacional e Logística da Marinha, SPOLM, CD-ROM, 16 p.
Valente, J. M.S. e Alves, R. A. F. S. (2005), Improved heuristics for the early/tardy scheduling problem with no idle time, *Computers & Operations Research*, 32, 557-569.
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*, 142, 271-281.
Ying, K. C. (2008), Minimizing earliness–tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm, *Computers & Industrial Engineering*, 55, 494-502.