SPOLM 2008

ISSN 2175-6295 Rio de Janeiro- Brasil, 05 e 06 de agosto de 2008.

UMA HEURÍSTICA HÍBRIDA PARA MINIMIZAR CUSTOS COM ANTECIPAÇÃO E ATRASO EM SISTEMAS DE PRODUÇÃO COM JANELAS DE ENTREGA E TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQÜÊNCIA

Marcone Jamilson Freitas Souza, Puca Huachi Vaz Penna
Universidade Federal de Ouro Preto
ICEB, Campus Universitário, CEP 35.400-000, Ouro Preto (MG)
marcone@iceb.ufop.br
puca@ouropreto.com.br

Frederico Augusto de Cezar Almeida Gonçalves
Centro Federal de Educação Tecnológica de Minas Gerais
Avenida Amazonas, 7675, CEP 30.510-000, Belo Horizonte (MG)
zepfred@gmail.com

Luiz Satoru Ochi
Universidade Federal Fluminense
Rua Passos da Pátria, 156 – Bloco E - 3° Andar, CEP 24.210-240, Niterói (RJ)
satoru@ic.uff.br

RESUMO

Este trabalho trata do problema de seqüenciamento em uma máquina, considerando a existência de janelas de entrega distintas e tempos de preparação da máquina dependentes da seqüência de produção, e tendo como objetivo minimizar os custos com a antecipação e o atraso da produção. Dada sua complexidade combinatória, propõe-se um algoritmo baseado em GRASP, Descida em Vizinhança Variável, Busca Tabu e Reconexão por Caminhos para resolvê-lo. O desempenho do algoritmo proposto é comparado com um da literatura e se mostra capaz de produzir soluções de melhor qualidade, tanto com respeito aos melhores resultados, quanto em relação aos resultados médios, por grupo de problemas-teste.

Palavras-Chaves: Seqüenciamento em uma máquina; GRASP; Busca Tabu; Descida em Vizinhança Variável; Reconexão por Caminhos.

ABSTRACT

This work deals with the single machine scheduling considering distinct due windows, sequence dependent setup and earliness/tardiness penalties. Due to combinatorial complexity, an algorithm based on GRASP, Variable Neighborhood Descent, Tabu Search and Path Relinking is proposed to solve it. The performance of the proposed algorithm is compared with one of the literature. The computational experiments show that the algorithm is able to produce solutions of better quality, both with respect to the best results, as to the average results, by each group of instances.

Keywords: Single Machine Scheduling; GRASP; Tabu Search; Variable Neighborhood Descent; Path Relinking;

1. INTRODUÇÃO

De acordo com França Filho (2007), o estudo de problemas de programação de tarefas envolvendo penalizações pela antecipação e pelo atraso é mais recente que aquele voltado para problemas em que o objetivo envolve uma função não-decrescente do instante de conclusão do processamento da tarefa, tais como tempo médio de fluxo, soma ponderada de atrasos e *makespan*. Para estes, custos mais elevados decorrem apenas do adiamento da conclusão das tarefas. Entretanto, com a filosofia *just-in-time* adotada por muitas empresas, o foco atual é penalizar também a conclusão das tarefas antes do instante em que elas são requeridas. Isto é justificado pelo fato de que concluir uma tarefa antecipadamente pode resultar em custos financeiros extras pela necessidade antecipada de capital e/ou espaço para armazenamento e/ou de outros recursos para manter e gerenciar o estoque.

Com relação às entregas, são encontrados na literatura três tipos de problemas: 1) os com datas de entrega comuns (*common due date*); 2) os com datas de entrega distintas (*distinct due dates*) e 3) os que têm janelas de entrega distintas (*distinct due windows*). No primeiro tipo, existe uma única data para entregar todas as tarefas, enquanto no segundo existe uma data de entrega específica associada a cada tarefa. No terceiro tipo, há um período para a conclusão de cada tarefa. Segundo Wan e Yen (2002), essa última situação aparece em situações de incertezas ou tolerâncias com relação às datas de entrega e não há custos se as tarefas forem concluídas dentro da janela de entrega.

Para problemas de produção envolvendo penalizações pela antecipação e pelo atraso com datas comuns de entrega, há muitas propriedades que podem ser consideradas nos algoritmos de solução e que permitem uma redução acentuada do esforço computacional na exploração do espaço de busca. Por exemplo, de acordo com Baker e Scuder (1990), na seqüência ótima, as tarefas que terminam seu processamento na data devida ou antes são ordenadas de forma não-crescente pela relação entre o tempo de processamento e a penalização pela antecipação; enquanto as tarefas que começam seu processamento na data devida ou depois são seqüenciadas de forma não-decrescente pela relação entre o tempo de processamento e a penalização pelo atraso. Em função disso, diz-se que a seqüência é V-*shaped*. Outra propriedade interessante é que na seqüência ótima não há tempos ociosos entre duas tarefas consecutivas.

Para o caso em que as datas de entrega são distintas, assim como para o caso em há janelas de entrega distintas, tais propriedades não são necessariamente válidas, tornando mais complexa a exploração do espaço de busca. Para esses tipos de problemas é necessário saber se é permitida a ociosidade das máquinas ou não. Segundo França Filho (2007), existem situações em que a ociosidade não é permitida por gerar custos maiores que aqueles decorrentes da conclusão antecipada das tarefas. Entretanto, há situações em que vale a pena manter uma máquina ociosa, mesmo que haja uma tarefa em condições de ter seu processamento iniciado. Para ilustrar esta situação, sejam duas tarefas *i* e *j*, que devem ser consecutivas em uma seqüência de produção envolvendo uma máquina, às quais estão associados os custos unitários por atraso de 100 e 20, respectivamente, custos unitários por antecipação de 22 e 2, e que tenham como datas de entrega os instantes 50 e 55, respectivamente, e 15 e 25 unidades de tempo de processamento, respectivamente. Suponha que a máquina já esteja disponível desde o instante 20. Se a tarefa *i* começar no instante 20, será concluída no instante 35 e, portanto, haverá uma penalização igual a 330 (=15×22) pela

antecipação. Nesse caso, a tarefa *j* poderá começar somente no instante 35. Considerando seu tempo de processamento, ela será concluída no instante 60, portanto 5 unidades de tempo atrasada em relação à data de entrega, resultando em um custo por atraso igual a 100 (=5×20), e um custo total de 430 (=330+100). Considere, agora, que o processamento da tarefa *i* seja iniciado no instante 35, mesmo sabendo que a máquina está disponível desde o instante 20. Neste caso, a tarefa *i* seria concluída no instante 50, sem multa nem por antecipação nem por atraso. A tarefa *j*, então, já poderia começar, terminando no instante 75, com 20 unidades de tempo de atraso, resultando em uma multa por atraso de 400 (=20×20). O custo total nesta última situação seria de 400 unidades, menor que aquele no qual a tarefa *i* começava desde o instante de liberação da máquina. Assim, não havendo restrições à ociosidade das máquinas, determinar a melhor data para iniciar o processamento de cada tarefa ou, equivalentemente, inserir tempos ociosos entre tarefas, poderá produzir soluções de menores custos.

Com relação à influência do tempo de preparação das tarefas na seqüência de produção, chamado de tempo de *setup*, observa-se que a maioria dos trabalhos científicos considera que tais tempos são negligenciáveis ou, então, podem ser adicionados aos tempos de processamento das tarefas. Isto é, considera-se que os tempos de *setup* são independentes da seqüência de tarefas. Contudo, muitos estudos, como os de Panwalkar (1973), *apud* Gupta e Smith (2006), indicam que em muitos processos produtivos, o tempo de preparação é dependente da seqüência de produção. De acordo com Kim e Bobrowski (1994), para modelar melhor casos práticos, esses tempos devem ser considerados explicitamente sempre que eles forem significativamente maiores que os tempos de processamento. Uma revisão de trabalhos sobre seqüenciamento da produção com tempos de preparação pode ser encontrada em Allahverdi *et al*. (1999).

Dentre os problemas de produção, o de seqüenciamento em uma máquina com minimização das penalidades por antecipação e atraso da produção (*Single Machine Scheduling for Minimizing Earliness and Tardiness Penalties*), representado simplesmente por PSUMAA, é um dos mais simples.

Além da importância prática desse problema, com inúmeras aplicações industriais (ALLAHVERDI *et al*., 1999), trata-se de um problema difícil de ser resolvido na otimalidade em tempos computacionais aceitáveis, por pertencer à classe NP-difícil (DU e LEUNG, 1990; LIAW, 1999; WAN e YEN, 2002). Essa conjunção de aplicabilidade e dificuldade de resolução na otimalidade tem motivado os pesquisadores a desenvolverem algoritmos eficientes para resolvê-lo.

Neste trabalho, trata-se o PSUMAA com janelas de entrega distintas e tempos de preparação da máquina dependentes da seqüência da produção. Surpreendentemente, essa variante do PSUMAA ainda não tem merecido a devido atenção. O único trabalho encontrado na literatura, com essas características, é o de Gomes Jr. *et al*. (2007).

Para resolvê-lo, propõe-se um algoritmo de 3 fases, combinando os procedimentos GRASP (Feo e Resende, 1995), *Variable Neighborhood Descent* – VND (Mladenovic e Hansen, 1997), Busca Tabu (Glover, 1986) e Reconexão por Caminhos (*Path Relinking*) (Glover, 1996). Na primeira fase é utilizado um procedimento baseado em GRASP para gerar uma solução inicial. Nesta fase, cada solução construída é refinada por um procedimento baseado em Descida em Vizinhança Variável (VND), que utiliza movimentos de realocação e troca de tarefas para explorar o espaço de soluções. Na segunda fase, a solução proveniente do GRASP é refinada por um procedimento Busca Tabu. Durante essa fase é formada uma lista de soluções elite, contendo soluções de alta qualidade e diferenciadas entre si. Como pós-otimização, na terceira e última fase, é aplicada a Reconexão por Caminhos a pares de soluções do grupo elite.

O restante deste trabalho está estruturado como segue. Na Seção 2 é feita uma apresentação de trabalhos correlatos. As características do problema estudado são detalhadas na Seção 3, enquanto na Seção 4 é descrito o algoritmo desenvolvido para resolução do PSUMAA. Na Seção 5 são apresentados e discutidos os resultados computacionais. A Seção 6 conclui o trabalho e aponta perspectivas futuras para melhoramento do algoritmo proposto.

2. TRABALHOS RELACIONADOS

Li (1997) e Liaw (1999) trataram o PSUMAA com datas comuns de entrega, sem permitir tempo ocioso de máquina. O primeiro autor decompõe o problema em dois subproblemas com uma estrutura mais simples, de forma que o limite inferior do problema é a soma dos limites inferiores desses dois subproblemas. O limite inferior de cada subproblema é determinado por relaxação lagrangiana. Um algoritmo *branch-and-bound* baseado é apresentado e usado para resolver instâncias de até 50 tarefas, dobrando a dimensão de problemas que podiam ser resolvidos na otimalidade com algoritmos exatos até aquela data. O autor propôs, também, procedimentos heurísticos baseados em busca local para resolver problemas de dimensões mais elevadas. No segundo trabalho é apresentado um algoritmo *branch-and-bound* que faz uso de procedimentos para determinar limites inferiores e superiores fortes. Regras de dominância são usadas para tentar eliminar nós não promissores na árvore de busca. É analisado o desempenho do algoritmo para resolver problemas de até 50 tarefas.

Feldmann e Biskup (2003) resolveram o PSUMAA com datas comuns de entrega por meio de três métodos: um algoritmo evolutivo, *Simulated Annealing* e uma versão aperfeiçoada da heurística *Threshold Accepting*, sendo esta última uma variante de *Simulated Annealing*. O mesmo problema foi tratado por Hino *et al*. (2005), os quais apresentaram métodos baseados nas metaheurísticas Busca Tabu e Algoritmos Genéticos, bem como hidridizações destas.

Ying (2008) trata do PSUMAA com datas comuns de entrega das tarefas e com tempo de *setup* de cada tarefa incluído em seu tempo de processamento e independente da seqüência de produção. O problema é resolvido pelo algoritmo *Recovering Beam Search* (RBS), que é uma versão aperfeiçoada do algoritmo *Beam Search* (BS). Este, por sua vez, consiste em um algoritmo *branch-and-bound* em que somente os *w* nós mais promissores de cada nível da árvore de busca são retidos para ramificação futura, enquanto os nós restantes são podados permanentemente. Para evitar decisões equivocadas com respeito à poda de nós que conduzam à solução ótima, o algoritmo RBS utiliza uma fase de recobrimento que busca por soluções parciais melhores que dominem aquelas anteriormente selecionadas.

Rabadi *et al*. (2004) focaram no PSUMAA com datas comuns de entrega e tempo de preparação da máquina dependente da seqüência de produção. Os autores apresentam um procedimento *branch-and-bound* que é capaz de resolver na otimalidade, em tempos aceitáveis, problemas-teste de até 25 tarefas, o que representava um avanço porque até aquela data algoritmos exatos para essa classe de problemas eram capazes de resolver somente problemas-teste de até 8 tarefas.

Chang (1999) tratou o PSUMAA com datas distintas de entrega por meio de um algoritmo *branch-and-bound*. É analisado o desempenho desse algoritmo para resolver problemas com até 45 tarefas. Também é desenvolvido um esquema para delimitar o cálculo de diferentes limites inferiores baseados no procedimento de eliminação de sobreposição de uma seqüência *just in time*. Propriedades e teoremas sobre procedimento de eliminação de sobreposição são apresentados.

Lee e Choi (1995) aplicaram Algoritmos Genéticos (AG) ao PSUMAA com datas de entrega distintas. Para determinar a data ótima de conclusão de processamento de cada tarefa

da seqüência produzida pelo AG, desenvolveram um algoritmo específico, de complexidade polinomial, que explora as características do problema.

Wan e Yen (2002) apresentaram um método baseado na metaheurística Busca Tabu (BT) para resolver o PSUMAA com janelas de entrega distintas. Para cada seqüência de tarefas gerada pela Busca Tabu é acionado um procedimento de complexidade polinomial para determinar a data de conclusão ótima do processamento de cada tarefa da seqüência. Este último procedimento é adaptado daquele proposto em Lee e Choi (1995), em que se consideram janelas de entrega em lugar de datas de entrega.

Gupta e Smith (2006) propuseram dois métodos, um baseado em GRASP e outro na heurística *space-based search*, para a resolução do problema de seqüenciamento em uma máquina considerando a existência de data de entrega para cada tarefa e tempo de preparação de máquina dependente da seqüência de produção, tendo como objetivo a minimização do tempo total de atraso. O método GRASP proposto foi dividido em três fases: Construção, Refinamento e Reconexão por Caminhos. Segundo os autores, a contribuição está em uma nova função custo para a fase de construção, uma nova variação do método VND para a fase de refinamento e uma fase de Reconexão por Caminhos usando diferentes vizinhanças.

Hallah (2007) trata do PSUMAA com datas de entrega distintas, não sendo permitida a existência de tempo ocioso entre as tarefas. O autor propõe um método híbrido que combina heurísticas de busca local (regras de despacho, método da descida e *Simulated Annealing*) e um algoritmo evolucionário.

Gomes Jr. *et al*. (2007) desenvolveram um modelo de programação linear inteira mista para o PSUMAA com janelas de entrega e tempo de preparação dependente da seqüência de produção, sendo este modelo utilizado para resolver na otimalidade problemas de até 12 tarefas. Esta modelagem serviu para comparar os resultados obtidos por um método heurístico baseado em GRASP, ILS e VND, também proposto pelos autores. Para cada seqüência gerada pela heurística, é acionado um algoritmo para determinar a data ótima de conclusão do processamento de cada tarefa. Tal algoritmo foi adaptado de Wan e Yen (2002), e inclui no tempo de processamento de uma tarefa o tempo de preparação da máquina, já que quando ele é acionado, já se conhece a seqüência de produção. O algoritmo desenvolvido pelos autores foi capaz de encontrar todas as soluções ótimas conhecidas.

3. O PROBLEMA DE SEQÜENCIAMENTO ABORDADO

O problema de seqüenciamento em uma máquina com penalidades por antecipação e atraso (PSUMAA) tratado neste trabalho tem as seguintes características: (*i*) Uma máquina deve processar um conjunto de *n* tarefas; (*ii*) A cada tarefa *i* está associado um tempo de processamento P_i e uma janela de entrega $[E_i, T_i]$, indicando uma data inicial E_i e uma data final T_i desejadas para o término de seu processamento. Se a tarefa *i* for finalizada antes de E_i há um custo α_i por unidade de tempo de antecipação. Caso a tarefa seja finalizada após T_i então há um custo β_i por unidade de tempo de atraso. As tarefas concluídas dentro da janela de entrega não proporcionam nenhum custo; (*iii*) A máquina pode executar no máximo uma tarefa por vez e uma vez iniciado o processamento de uma tarefa, não é permitida a sua interrupção; (*iv*) Todas as tarefas estão disponíveis para processamento na data 0; (*v*) Entre duas tarefas *i* e *j* consecutivas é necessário um tempo S_{ij} de preparação da máquina, chamado tempo de *setup*. Assume-se que a máquina não necessita de tempo de preparação inicial, ou seja, o tempo de preparação da máquina para o processamento da primeira tarefa na seqüência é igual a 0; (*vi*) É permitido tempo ocioso entre a execução de duas tarefas consecutivas.

4. METODOLOGIA

4.1. REPRESENTAÇÃO DE UMA SOLUÇÃO

Uma solução do PSUMAA é representada por um vetor v de n elementos, onde cada posição i = 1, 2, ..., n indica a ordem de produção da i-ésima tarefa da seqüência. Assim, na seqüência v = {5, 3, 2, 1, 4, 6}, a tarefa 5 é a primeira a ser realizada e a tarefa 6, a última.

4.2. 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 seqüência de produção, realocação de uma tarefa para outra posição na seqüência de produção e realocação de um bloco de k tarefas ($2 \leq k \leq n$-1). Esses movimentos definem, respectivamente, as estruturas de vizinhança N^T, N^R e N^{Or}. O terceiro movimento foi proposto originalmente em Or (1976) para explorar o espaço de soluções do Problema do Caixeiro Viajante. As realocações são feitas tanto para posições antecessoras quanto para posições sucessoras àquelas em que a tarefa ou bloco se encontram.

Dado um conjunto com n tarefas, há $n(n\text{-}1)/2$ vizinhos possíveis na vizinhança N^T. Por exemplo, a solução v' = {5, 4, 2, 1, 3, 6} é um vizinho da solução v na vizinhança N^T, pois é obtido pela troca da tarefa 3, que está na segunda posição de v, com a tarefa 4, que está na quinta posição de v. Na segunda estrutura de vizinhança, N^R, há $n(n\text{-}1)$ vizinhos possíveis. Por exemplo, a solução v' = {5, 2, 1, 4, 3, 6} é vizinha de v na vizinhança N^T, pois é obtida pela realocação da tarefa 3 para depois da tarefa 4 na seqüência de produção v. Na vizinhança N^{Or} há $n^2 - (2k\text{–}1)n + k(k - 1)$ vizinhos de k blocos. Por exemplo, v' = {2, 1, 5, 3, 4, 6} é um vizinho da solução v na vizinhança N^{Or}, pois o bloco formado pelas tarefas 5 e 3 foi movido de v para depois da tarefa 1.

4.3. FUNÇÃO DE AVALIAÇÃO

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

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

em que e_i e t_i (e_i , $t_i \geq 0$) indicam, respectivamente, o tempo de antecipação e atraso da tarefa i.

4.4. GERAÇÃO DA SOLUÇÃO INICIAL

A solução inicial é gerada por um procedimento GRASP (Feo e Resende, 1995). Como tal, há duas fases, construção e refinamento, aplicadas *GRASPMax* vezes, sendo retornado a melhor das soluções obtidas. A Figura 1 esquematiza esse procedimento.

```
procedimento GRASP-VND(v, γ, GRASPMax, MRDMax)
1    f* ← ∞;
2    Para i = 1 até GRASPMax faça
3        Escolha γ∈ Γ
4        v = Constroi_Solucao_GRASP(γ);
5        v' = VND1(v, MRDMax );
6        Se (f(v') < f*) então
7            v* ← v';  f* ← f(v');
8        fim-se;
9    fim-para;
10   v = VND2(v*, MRDMax );
11   Retorne v;
fim GRASP-VND;
```

Figura 1: Procedimento de geração de uma solução inicial

Na fase de construção (Linha 4 da Figura 1), uma solução é gerada gradativamente, sendo que a cada passo, é adicionada uma única tarefa à seqüência parcial. Para se escolher a tarefa a ser adicionada, é feita uma lista de candidatos (LC) com todas as tarefas ainda não seqüenciadas. As tarefas dessa lista são ordenadas pela data de início da janela de entrega de cada tarefa, sendo a com a data mais cedo (E_{min}) a primeira da lista e a com data mais tarde (E_{max}) a última, tal como na heurística EDD (*Earliest Due Date*). A partir dessa lista, é formada uma lista restrita de candidatos (LRC), com as tarefas melhor ranqueadas segundo o critério da data de início da janela de entrega das tarefas. O tamanho dessa lista restrita de candidatos é definido por um parâmetro γ, com $0 \leq \gamma \leq 1$, escolhido dentro de um conjunto Γ, como em Gomes Jr. *et al*. (2007). Fazem parte dessa lista todas as tarefas *i* cujas datas E_i sejam menores ou iguais à $E_{min} + \gamma (E_{max} - E_{min})$. A seguir, aleatoriamente é escolhida uma tarefa dessa lista e esta é adicionada à solução parcial. A construção é encerrada quando todas as tarefas forem alocadas.

A segunda fase do procedimento GRASP proposto (Linha 5 da Figura 1) consiste em refinar a solução pela heurística Descida em Vizinhança Variável (*Variable Neighborhood Descent* - VND) usando-se as duas estruturas de vizinhança, N^T e N^R, nesta ordem. São escolhidas aleatoriamente duas tarefas distintas e trocada a ordem de seus processamentos na seqüência. Aplica-se a seguir o algoritmo de datas ótimas. Se a nova seqüência produzir uma solução com um valor menor para a função de avaliação, a nova seqüência é aceita e passa a ser a solução corrente; caso contrário, é testada outra troca aleatória. A fase de troca pára quando houver *MRDMax* trocas consecutivas sem melhora na função de avaliação. Nesta última situação, passa-se a fazer movimentos de realocação. Para cada uma de todas as possíveis realocações, é feita a avaliação e aplicado o algoritmo de datas ótimas. Se for produzida uma solução global de melhor qualidade, interrompe-se esta fase e volta-se para a fase de troca; caso contrário prossegue-se com as realocações. O procedimento termina quando não houver melhora na solução global nem com movimentos de troca nem com movimentos de realocação. Observa-se que a solução resultante deste procedimento não é necessariamente um ótimo local com relação à vizinhança de troca, visto que nem toda essa vizinhança é explorada a cada iteração.

4.5. BUSCA TABU APLICADA AO PSUMAA

O procedimento Busca Tabu (*Tabu Search* – TS), (Glover, 1986), implementado é iniciado com uma solução inicial gerada pelo procedimento GRASP. A cada iteração todos os vizinhos desta solução são avaliados, sendo escolhido o melhor deles que não seja tabu ou, se tabu, satisfaça a condição de aspiração, no caso, de o vizinho gerar um valor melhor que o da melhor solução até então. A exploração da vizinhança é feita tendo por base as estruturas de vizinhança N^T e N^R, alternando-as a cada iteração, isto é, na primeira iteração explora-se o melhor vizinho com movimentos de troca, na segunda iteração com movimentos de realocação e assim por diante. O objetivo de se verificar se o movimento é tabu ou não, é evitar ciclos de soluções recentemente visitadas.

Quando um vizinho $v' \in N^T(v) \cup N^R(v)$ é escolhido, ele se torna a solução corrente, independentemente de ele ser melhor ou pior que a solução corrente e um atributo que caracterize esta solução é inserido em uma lista tabu *T*, proibindo o retorno a ele durante um prazo. Dada a troca entre uma tarefa *i* e uma tarefa *j*, o atributo que é inserido na lista tabu é a aresta (tarefa *j*, tarefa sucessora de *j* da solução modificada). No caso de realocação de uma tarefa *i* para antes ou depois de uma tarefa *j*, o atributo a ser inserido é a aresta (tarefa *i*,

sucessor da tarefa *i* na solução modificada) se a tarefa *j* não for a última e a aresta (tarefa antecessora a *i* na solução modificada, tarefa *i*) se for a última. A lista tabu é limitada a |*T*| elementos. Quando a lista tabu está completa, o elemento mais antigo é retirado da lista, e aquele recentemente obtido é inserido, tal como em uma fila (FIFO). Para ilustrar o funcionamento da lista tabu, considere a seqüência $v = \{5, 3, 2, 1, 4\}$ e o movimento de troca envolvendo as tarefas 3 e 1. O resultado desta troca produzirá o vizinho $v' = \{1, 3, 2, 5, 4\}$. Assim, é incluído na lista tabu o par formado pela subseqüência (tarefa 1, tarefa subseqüente a 1 na seqüência *v'*), isto é, $T \leftarrow T \cup (1, 3)$.

O procedimento é encerrado ao final de *TSmax* iterações sem melhora na solução global. O pseudocódigo do procedimento Busca Tabu implementado é apresentado na Figura 2, sendo a atualização do Grupo Elite (*GE*), linha 10 da Figura 2, feita de acordo com a Subseção 4.6.

```
procedimento TS(v, divElite, GE, TSMax)
 1   v* ← v;
 2   Iter ← 0;{Contador do número de iterações}
 3   MelhorIter ← 0; {Iteração mais recente que forneceu v*}
 4   T ← Ø; {Lista Tabu}
 5   GE ← Ø; {Grupo Elite, utilizado pelo procedimento Path Relinking}
 6   enquanto (Iter – MelhorIter ≤ TSMax) faça
 7       Iter ← Iter + 1;
 8       Seja v' ← v ⊕ m o melhor elemento de V ⊆ N(v) (N=N^T se Iter for ímpar e
         N=N^R se Iter for par, tal que o movimento m não seja tabu (m ∉ T) ou v' atenda
         a condição de aspiração f(v') < f(v*);
 9       Atualize a Lista Tabu T e o Grupo Elite GE;
 10      v ← v';
 11      se f(v) < f(v*) então
 12          v* ← v; MelhorIter ← Iter;
 13      fim-se;
 14  fim-enquanto;
 15  v ← v*;
 16  Retorne v;
fim TS;
```

Figura 2: Procedimento *Busca Tabu*

4.6. RECONEXÃO POR CAMINHOS

Reconexão por Caminhos (*Path Relinking* – PR), proposto por Glover (1996), é uma estratégia de intensificação normalmente aplicada ou como pós-otimização ou como refinamento de ótimos locais obtidos durante a busca. Dado um par de soluções, o objetivo da técnica é partir de uma delas, dita solução base, e chegar à outra, dita solução guia, por meio da adição na solução base de atributos da solução guia.

No algoritmo proposto, a Reconexão por Caminhos trabalha em conjunto com a Busca Tabu (*Tabu Search* – TS), sendo utilizada como ferramenta de pós-refinamento aplicada após a execução da Busca Tabu. Para tanto, durante a exploração do espaço de busca pelo procedimento TS, a PR utiliza um conjunto de soluções, conhecido como Grupo Elite (*GE*). Para fazer parte desse grupo, cada solução candidata deve satisfazer a um dos dois seguintes critérios: 1) ser melhor que a melhor das |*GE*| soluções do grupo elite; 2) ser melhor que a pior das |*GE*| soluções do grupo elite e se diferenciar delas de determinado percentual dos atributos, dado por *divElite*. O atributo considerado é o par de tarefas *i* e *j* consecutivas. Assim, por exemplo, dadas as seqüências $v_1 = \{1, 4, 3, 2, 5, 6\}$ e $v_2 = \{5, 3, 2, 1, 4, 6\}$, elas

têm 40% de atributos iguais, a saber, as subseqüências (1, 4) e (3, 2). O objetivo desta estratégia é evitar a inclusão no Grupo Elite de soluções muito parecidas. Estando o grupo elite já formado, quando uma solução entra, a de pior avaliação sai.

O procedimento PR foi implementado como segue. Para cada par de soluções elite, caminha-se de forma bidirecional, isto é, tanto da pior solução para a melhor (dita Reconexão por Caminhos Progressiva – *Forward Path Relinking*), quanto da melhor para a pior solução (dita Reconexão por Caminhos Regressiva – *Backward Path Relinking*).

Determinadas as soluções base e guia, o procedimento é executado inserindo-se gradativamente um atributo da solução guia na solução base. Por exemplo, dada a solução guia = {5, 3, 2, 1, 4, 6}, a cada iteração do procedimento insere-se um par de tarefas consecutivas dessa solução, a saber, (5, 3), (3, 2), (2, 1), (1, 4), (4, 6), na solução base. Cada par é inserido na ordem em que aparece na solução guia. Após cada inserção, as tarefas da solução base que foram substituídas saem da posição que ocupavam e assumem as posições daquelas que entraram. A solução base sofre, então, uma busca local (no caso, uma descida randômica usando os movimentos de troca e realocação, tal como descrito na Subseção 4.4) onde se fixam os atributos da solução guia que já foram incorporados à solução base. A cada iteração, a solução base recebe todos os atributos ainda não incorporados da solução guia, um de cada vez, gerando uma nova seqüência para cada atributo específico da solução guia. Ao final da iteração, o atributo inserido que produziu a melhor solução após a busca local é incorporado em definitivo à solução base, se tornando fixo. O procedimento é encerrado quando se alcança a solução guia, isto é, quando a solução base passa a ter todos os atributos da solução guia. Para mostrar o funcionamento do procedimento, considere como solução guia a seqüência {5, 3, 2, 1, 4, 6} e como solução base {2, 6, 5, 1, 4, 3}.

Passo 1: Tem como objetivo inserir, na solução base, cada um dos 5 pares de sub-seqüências da solução guia, {(5, 3), (3, 2), (2, 1), (1, 4), (4, 6)}, na ordem em que aparecem e verificar qual inserção traz o melhor resultado. Para tanto, procede-se como segue:

Passo 1.1) Inserir as tarefas 5 e 3 do par (5, 3) nas duas primeiras posições da solução base. Para tanto, a tarefa 5 ocupa a primeira posição da solução base, onde está a tarefa 2. Esta, por sua vez, sai da primeira posição e vai para a terceira posição, onde está a tarefa 5, resultando na seqüência {5, 6, 2, 1, 4, 3}. A seguir, é inserida a tarefa 3 na segunda posição da seqüência base. Como nesta posição está a tarefa 6, esta se desloca para a última posição, onde está localizada a tarefa 3. Após esta inserção obtém-se a seqüência {5, 3, 2, 1, 4, 6}. Sobre esta seqüência é aplicada busca local (VND com Descida Randômica com movimentos de troca e Descida completa com movimentos de realocação, descrita na Subseção 4.4), proibindo-se qualquer movimentação das tarefas da solução guia que foram inseridas, no caso, 5 e 3.

Passo 1.2) Inserir as tarefas 3 e 2 do par (3, 2) na segunda e terceira posição, respectivamente, da solução base. Para tanto, a tarefa 3 ocupa a segunda posição da solução base, onde está a tarefa 6. Esta, por sua vez, sai da segunda posição e vai para a última posição, onde está a tarefa 3, resultando na seqüência {2, 3, 5, 1, 4, 6}. A seguir, é inserida a tarefa 2 na terceira posição da seqüência base. Como nesta posição está a tarefa 5, esta se desloca para a primeira posição, onde está localizada a tarefa 2. Após esta inserção obtém-se a seqüência {5, 3, 2, 1, 4, 6}. Sobre esta seqüência é aplicada busca local, tal como anteriormente, proibindo-se qualquer movimentação das tarefas 3 e 2.

...

Passo 1.5) Inserir as tarefas 2 e 1 do par (2, 1) na quinta e sexta posição, respectivamente, da solução base e prosseguir como no passo anterior.

A inserção que produzir o melhor resultado para a função de avaliação, dentre os

passos 1.1 a 1.5, é realizada. Após a inserção é feita busca local por um procedimento denominado VND_3, no qual é aplicada Descida Randômica usando movimentos de troca seguida de Descida Completa com movimentos de realocação. Neste procedimento, sempre que a Descida Completa com movimentos de realocação produz uma solução de melhora, retorna-se à Descida Randômica com movimentos de troca. Durante a aplicação do VND_3 é proibida qualquer movimentação das tarefas da solução guia que foram inseridas. Por exemplo, se a seqüência do Passo 1 resultar em melhor valor para a função de avaliação, as tarefas 5 e 3 não podem movimentar-se. Assim, ter-se-á inserido na solução base um atributo da solução guia. A inserção dos demais atributos é feita de forma semelhante, adicionando-se o fato de que durante a busca local, não somente o par recentemente inserido permanece fixo, mas também todos os demais pares de subseqüências da solução guia já inseridos.

Sempre que o procedimento PR gera uma solução melhor que a melhor do grupo elite, a inserção de atributos é interrompida, o grupo elite é automaticamente atualizado e o processo se inicia novamente. Se encerradas as inserções dos atributos, o procedimento tiver gerado uma solução melhor que a pior, satisfazendo aos critérios de diversidade estabelecidos anteriormente, o grupo elite é atualizado com a saída de seu pior elemento. O procedimento termina após *PRMax* aplicações da Reconexão por Caminhos sem alteração na melhor solução. Neste caso, retorna-se a melhor solução encontrada, bem como o conjunto elite atualizado.

4.7. ALGORITMO GTSPR

O algoritmo proposto, GTSPR, é de 3 fases e combina características dos procedimentos GRASP e VND (Subseção 4.4), Busca Tabu (Subseção 4.5) e Reconexão por Caminhos (Subseção 4.6). Seu pseudocódigo é apresentado na Figura 3.

Algoritmo GTSPR(*v, γ, divElite*, |*T*|, *GRASPMax*, *MRDMax*, *TSMax*, *PRMax*)
1 $v_0 \leftarrow$ *GRASP-VND*(*v, γ, GRASPMax*, *MRDMax*);
2 $v_1 \leftarrow$ *TS*(v_0, |T|, *divElite*, *GE*, *TSMax*, *MRDMax*);
3 $v \leftarrow$ *PR* (*GE*, *divElite*, *PRMax*);
4 Retorne *v*;
fim GTSPR;

Figura 3: Algoritmo GTSPR proposto

5. RESULTADOS COMPUTACIONAIS

Os problemas-teste utilizados foram os mesmos de Gomes Jr. *et al*. (2007), com número de tarefas igual a 8, 9, 10, 11, 12, 15, 20, 25, 30, 40, 50 e 75. Esta base de dados foi gerada por estes autores usando procedimentos pseudo-aleatórios, considerando distribuição uniforme e parâmetros estabelecidos em Wan e Yen (2002), Liaw (1999) e Mazzini e Armentano (2001). Os custos por atraso da produção (β_i) são números inteiros no intervalo [20, 100], enquanto os custos por antecipação da produção (α_i) são *k* vezes o custo por atraso da mesma tarefa, sendo *k* um número real aleatório no intervalo [0, 1]. Os tempos de processamento P_i das tarefas são números inteiros entre 1 e 100. Já os centros das janelas de entrega são inteiros pertencentes ao intervalo [(1-*TF*-*RDD*/2)/*MS*, (1-*TF*+*RDD*/2)/*MS*], sendo *MS* o tempo total de processamento de todas as tarefas, *TF* o fator de atraso e *RDD* a variação relativa da janela de entrega. Os valores considerados para *TF* foram 0,1; 0,2; 0,3 e 0,4 enquanto que para *RDD* os valores foram 0,8; 1,0 e 1,2. Os tamanhos das janelas de entrega são números inteiros no intervalo [1, *MS*/*n*]. Os tempos de preparação da máquina (S_{ij}) são inteiros no intervalo [0, 50]. Nesta base de dados os tempos de preparação são simétricos, ou

seja, $S_{ij}= S_{ji}$. Como *TF* possui quatro valores diferentes e *RDD* três valores diferentes, há 12 problemas-teste para cada número de tarefas, totalizando 144 problemas-teste.

O algoritmo proposto foi desenvolvido na linguagem C++, utilizando o compilador Borland C++ Builder 5.0. Todos os experimentos foram realizados em um computador Pentium Core 2 Quad (Q6600) 2,4 GHz com 4 GB de memória RAM e sistema operacional Windows XP. Apesar de o processador deste equipamento possuir 4 núcleos, o algoritmo desenvolvido não foi otimizado para multiprocessamento. Os parâmetros do algoritmo são apresentados na Tabela 1, em que *n* é o número de tarefas. Cada problema-teste foi executado 30 vezes.

Tabela 1: Parâmetros do algoritmo GTSPR

Parâmetro	**Valor**
Γ(conjunto de parâmetros γ da fase de construção GRASP)	0; 0,02; 0.04; 0.12 e 0.14
GRASPMax (Número máximo de iterações GRASP)	6
MRDMax (Número máximo de iterações sem melhora da busca local)	$7 \times n$
TSMax (Número máximo de iterações sem melhora da Busca Tabu)	$2 \times n$
PRMax (N° máx. de aplicações sem melhora da Reconexão por Caminhos)	2
$\|T\|$ (Tamanho máximo da Lista Tabu)	$2 \times n$
$\|GE\|$ (Tamanho máximo do Grupo Elite)	5
divElite (% mínimo de diversificação entre os membros do Grupo Elite)	0,4

O algoritmo proposto foi comparado com o de Gomes Jr. *et al.* (2007) por meio de duas medidas, dadas pelas fórmulas (2) e (3):

$$imp_i^{best} = \frac{f_i^{GJr} - f_i^{GTSPR}}{f_i^{GJr}} \qquad (2) \qquad imp_i^{avg} = \frac{\bar{f}_i^{GJr} - \bar{f}_i^{GTSPR}}{\bar{f}_i^{GJr}} \qquad (3)$$

em que, para cada problema-teste *i* do grupo, f_i^{GTSPR} representa o melhor valor encontrado pelo algoritmo proposto, f_i^{GJr} é o melhor valor encontrado por Gomes Jr. *et al*. (2007), $\bar{f}_i^{GTSPR}$ é o valor médio das soluções finais produzidas pelo algoritmo proposto e $\bar{f}_i^{GJr}$ é valor médio das soluções finais encontradas por Gomes Jr. *et al*. (2007).

Na Tabela 2 são comparados os melhores resultados e os resultados médios produzidos pelos algoritmos em questão. Nas colunas intituladas *Melhora* são apresentados os percentuais de melhora do algoritmo proposto sobre o algoritmo de Gomes Jr. *et al*. (2007), resultados da aplicação das fórmulas (2) e (3). As duas últimas colunas mostram o tempo de processamento do algoritmo de Gomes Jr. *et al.* (2007) e do algoritmo proposto. Observa-se que o primeiro foi testado em uma máquina diferente, no caso, um microcomputador Athlon XP 64 Bits 3000+ (aproximadamente 2 GHz) com 1 GB de RAM. Os resultados dos problemas-teste envolvendo até 12 tarefas não foram apresentados por questão de espaço. Contudo, os dois algoritmos alcançaram todas as soluções ótimas conhecidas, tendo o primeiro uma variabilidade média de 0,21%, no máximo, e o segundo, de 0,09%, no máximo.

Pela Tabela 2, observa-se que todas as melhores soluções ou foram alcançadas ou superadas em até 5,58%, na média, por grupo. A exceção ocorreu para o problema-teste INST5011, na qual houve uma piora de 1,69% na qualidade da solução final. Entretanto, em compensação, o desempenho médio foi melhor, com uma melhora de 3,91%. Com relação ao desempenho médio nos demais casos, houve uma melhora de até 8,52%, na média por grupo e de até 20,96% se considerado cada problema-teste individualmente. O desempenho médio foi inferior apenas em 14 dos problemas-teste, mas limitado a um máximo de 1,41%.

Tabela 2: Comparação de resultados algoritmo GTSPR × algoritmo Gomes Jr. *et al.* (2007)

Problema -teste	Melhor Valor			Valor Médio			Tempo (s)	
	Gomes Jr. *et al.* (2007)	GTSPR	Melhora %	Gomes Jr. *et al.* (2007)	GTSPR	Melhora %	Gomes Jr.	GTSPR
INST1501	18.276	18.276	0,00	18.276	18.276	0,00%	0,64	0,37
INST1502	19.622	19.622	0,00	20.198	20.156	0,21	1,23	0,54
INST1503	11.505	11.505	0,00	11.952	11.843	0,91	1,09	0,48
INST1504	15.164	15.164	0,00	15.250	15.250	0,00	1,00	0,49
INST1505	12.924	12.924	0,00	13.141	12.991	1,14	0,79	0,41
INST1506	9.396	9.396	0,00	9.423	93.96	0,29	0,60	0,34
INST1507	46.544	46.544	0,00	46.977	47.123	-0,31	1,23	0,52
INST1508	24.899	24.899	0,00	26.201	26.570	-1,41	1,19	0,49
INST1509	14.457	14.457	0,00	14.457	14.457	0,00	0,56	0,33
INST1510	33.128	33.128	0,00	33.556	33.209	1,03	1,11	0,54
INST1511	42.522	42.522	0,00	42.866	42.522	0,80	1,12	0,54
INST1512	12.793	12.793	0,00	12.793	12.793	0,00	0,68	0,43
Média			0,00			0,22	0,94	0,46
INST2001	16.294	16.294	0,00	16.294	16.294	0,00	1,95	1,30
INST2002	18.107	18.107	0,00	18.660	18.486	0,93	3,63	1,49
INST2003	20.249	20.249	0,00	20.461	20.449	0,06	3,07	1,48
INST2004	29.941	29.941	0,00	30.283	29.960	1,07	4,43	2,02
INST2005	59.158	59.158	0,00	59.736	59.482	0,43	4,60	2,43
INST2006	31.053	31.053	0,00	32.456	32.265	0,59	4,86	2,18
INST2007	40.438	40.438	0,00	42.005	41.078	2,21	4,84	2,39
INST2008	29.186	29.186	0,00	29.368	29.244	0,42	3,18	1,61
INST2009	67.827	67.827	0,00	69.246	68.797	0,65	5,61	2,77
INST2010	34.283	34.283	0,00	34.656	34.691	-0,10	5,52	2,38
INST2011	54.538	54.538	0,00	54.911	55.145	-0,43	5,90	2,20
INST2012	79.599	79.599	0,00	80.171	79.791	0,47	4,58	2,32
Média			0,00			0,52	4,35	2,05
INST2501	23.397	23.397	0,00	23.480	23.403	0,33	13,27	6,38
INST2502	48.540	48.540	0,00	48.801	49.049	-0,51	13,24	6,38
INST2503	18.503	18.503	0,00	18.526	18.563	-0,20	9,32	4,61
INST2504	25.645	25.645	0,00	25.785	25.702	0,32	13,44	6,52
INST2505	35.865	35.865	0,00	36.796	36.128	1,82	14,14	7,46
INST2506	14.189	14.189	0,00	14.219	14.192	0,19	7,85	4,05
INST2507	37.313	37.313	0,00	37.857	37.406	1,19	15,26	5,91
INST2508	44.638	44.638	0,00	45.150	44.671	1,06	13,23	7,23
INST2509	12.839	12.839	0,00	12.874	12.839	0,27	10,52	6,64
INST2510	51.415	51.415	0,00	57.174	55.889	2,25	18,25	9,06
INST2511	44.808	44.808	0,00	47.610	47.581	0,06	15,87	7,64
INST2512	34.197	34.197	0,00	35.255	35.163	0,26	15,10	7,52
Média			0,00			0,59	13,29	6,62
INST3001	43.673	43.673	0,00	44.778	45.039	-0,58	38,91	16,78
INST3002	41.983	41.983	0,00	42.111	42.014	0,23	39,48	17,20
INST3003	21.951	21.951	0,00	22.375	22.183	0,86	24,60	12,85
INST3004	64.943	64.943	0,00	68.138	67.704	0,64	57,49	23,56
INST3005	74.100	74.100	0,00	76.040	75.471	0,75	46,12	20,16
INST3006	29.829	29.829	0,00	30.832	30.875	-0,14	33,51	16,18
INST3007	74.336	74.336	0,00	75.378	75.301	0,10	41,04	19,86
INST3008	69.770	69.770	0,00	74.097	74.544	-0,60	53,20	22,99
INST3009	21.335	21.335	0,00	21.641	21.459	0,84	22,73	13,11
INST3010	73.702	73.702	0,00	77.065	76.595	0,61	48,36	22,62
INST3011	35.190	35.190	0,00	38.152	36.036	5,55	31,09	18,03
INST3012	83.976	83.976	0,00	86.008	85.565	0,52	44,30	20,53
Média			0,00			0,73	40,07	18,66

Tabela 2: Continuação

Problema -teste	Melhor Valor			Valor Médio			Tempo (s)	
	Gomes Jr. *et al.* (2007)	GTSPR	Melhora %	Gomes Jr. *et al.* (2007)	GTSPR	Melhora %	Gomes Jr.	GTSPR
INST4001	57.086	57.086	0,00	57.751	58.107	-0,62	160,29	78,42
INST4002	49.306	49.306	0,00	49.705	49.528	0,36	108,29	61,42
INST4003	28.643	28.643	0,00	28.834	28.675	0,55	24,42	19,13
INST4004	99.871	99.828	0,04	102.171	102.958	-0,77	210,85	95,83
INST4005	29.863	29.863	0,00	30.594	30.956	-1,18	96,74	57,72
INST4006	32.303	32.303	0,00	33.403	33.142	0,78	83,50	52,42
INST4007	117.047	116.479	0,49	125.443	121.904	2,82	186,75	109,40
INST4008	44.847	44.847	0,00	46.653	46.362	0,62	122,02	79,18
INST4009	77.378	77.378	0,00	82.618	81.192	1,73	189,65	98,05
INST4010	93.591	93.225	0,39	97.763	97.140	0,64	230,40	114,04
INST4011	106.470	105.285	1,11	119.856	114.819	4,20	202,31	125,77
INST4012	127.269	127.269	0,00	135.908	136.443	-0,39	254,26	118,54
Média			0,17			0,73	155,79	84,16
INST5001	105.563	105.426	0,13	111.204	111.677	-0,43	585,60	331,95
INST5002	69.631	69.244	0,56	73.810	73.116	0,94	258,97	199,00
INST5003	60.259	60.259	0,00	61.219	60.749	0,77	233,27	185,51
INST5004	96.858	96.837	0,02	101.444	100.998	0,44	529,48	314,74
INST5005	80.597	80.140	0,57	82.473	81.487	1,20	323,06	209,68
INST5006	64.500	64.500	0,00	65.746	65.079	1,01	364,45	184,80
INST5007	129.284	125.350	3,04	140.856	135.012	4,15	680,62	436,32
INST5008	77.165	76.565	0,78	80.279	78.686	1,98	426,34	235,92
INST5009	55.213	54.234	1,77	60.435	59.434	1,66	584,56	316,49
INST5010	171.912	162.543	5,45	181.267	179.638	0,90	700,17	468,13
INST5011	176.332	179.319	-1,69	196.956	189.262	3,91	739,21	464,21
INST5012	83.686	83.686	0,00	93.811	91.939	2,00	481,63	316,57
Média			0,89			1,54	492,28	305,28
INST7501	258.776	253.362	2,09	277.913	265.920	4,32	1.809,4	3.638,6
INST7502	107.779	106.990	0,73	115.331	111.134	3,64	906,6	1.926,8
INST7503	85.960	85.514	0,52	92.820	88.121	5,06	238,1	1.106,2
INST7504	311.870	306.261	1,80	324.516	312.997	3,55	1.416,5	3.537,0
INST7505	116.346	115.367	0,84	130.908	124.897	4,59	1.201,7	2.727,2
INST7506	142.313	139.934	1,67	152.408	143.078	6,12	608,4	1.297,1
INST7507	332.419	318.107	4,31	369.624	331.189	10,40	1.876,1	5.759,8
INST7508	218.812	193.600	11,52	255.181	224.370	12,07	1.838,5	4.383,4
INST7509	202.863	177.743	12,38	224.472	207.876	7,39	1.417,6	3.058,4
INST7510	359.904	323.506	10,11	392.750	351.642	10,47	1.889,0	5.965,9
INST7511	221.263	198.278	10,39	274.439	216.909	20,96	1.673,9	4.552,5
INST7512	137.306	122.721	10,62	155.229	134.074	13,63	1.541,3	3.714,2
Média			5,58			8,52	1.368,1	3.472,3

Com relação ao tempo de processamento, apesar de uma comparação justa não ser possível, devido ao fato de as máquinas usadas para teste terem sido diferentes, o algoritmo proposto teve desempenho satisfatório. A exceção ocorreu nos problemas-teste envolvendo 75 tarefas, o qual demandou um tempo bem mais elevado. Entretanto, nesta classe de problemas, houve, em contrapartida, uma melhora significativa na qualidade da melhor solução (de 5,58%, na média), bem como no desempenho médio, com uma melhora média de 8,52%.

6. CONCLUSÕES

Neste trabalho foi tratado o Problema de Seqüenciamento em uma Máquina com penalidades por Antecipação e Atraso da produção (PSUMAA), considerando janelas de entrega distintas e tempo de preparação da máquina dependente da seqüência de produção.

Um algoritmo heurístico de três fases, combinando características dos procedimentos

GRASP, VND, Busca Tabu e Reconexão por Caminhos, foi proposto. A exploração do espaço de soluções é feita considerando-se movimentos de realocação e troca de tarefas, bem como realocação de blocos de tarefas. Na primeira fase, é construída uma solução pela aplicação do procedimento GRASP, tendo-se o VND como método de busca local. Esta solução inicial é, a seguir, refinada por um procedimento Busca Tabu, armazenando-se, durante a busca, um grupo de soluções elite. Encerrada a Busca Tabu, na terceira e última fase, é aplicada a Reconexão por Caminhos a todos os pares de soluções deste grupo elite.

O algoritmo proposto foi aplicado a problemas-teste envolvendo até 75 tarefas. Os resultados computacionais mostraram sua superioridade em relação ao de Gomes Jr. *et al*. (2007), quando considerados os melhores resultados e os resultados médios, por grupo de problemas-teste. Como trabalho futuro aponta-se o estudo de propriedades que possam ser aplicadas ao problema, de forma a reduzir o espaço de busca e, conseqüentemente, o tempo de processamento do algoritmo.

7. AGRADECIMENTOS

Agradece-se à FAPERJ, processo E-26/101.023/2007, pelo apoio concedido.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] **Allahverdi A., Gupta J. N. D. e Aldowaisan T.** (1999), A review of scheduling research involving setup considerations, *OMEGA*, v. 27, p. 219-239.

[2] **Baker, K. R. e Scudder, G. D.** (1990), Sequencing with Earliness and Tardiness Penalties: A Review, *Operations Research*, v. 38, p. 22–36.

[3] **Chang, P. C.** (1999), A Branch and Bound Approach for Single Machine Scheduling with Earliness and Tardiness Penalties, *Computers and Mathematics with Applications*, v. 37, n. 10, p. 133–144.

[4] **Du, J. e Leung, J. Y. T.** (1990), Minimizing total tardiness on one machine is NP-hard, *Mathematics of Operations Research*, v. 15, p. 483-495.

[5] **Feldmann, M. e Biskup, D.** (2003), Single-Machine Scheduling for Minimizing Earliness and Tardiness Penalties by Meta-heuristic Approaches, *Computers and Industrial Engineering*, v. 44, p. 307–323.

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

[7] **França Filho, M. F.** (2007), GRASP e Busca Tabu aplicados a problemas de programação de tarefas em máquinas paralelas, Tese de doutorado, Campinas: Departamento de Engenharia de Sistemas, UNICAMP.

[8] **Glover, F.** (1986), Future paths for Integer Programming and links to Artificial Intelligence, *Computers and Operations Research*, v. 5, p. 553–549.

[9] **Glover, F.** (1996), Tabu search and adaptive memory programming – Advances, applications and challenges, *In* Barr RS, Helgason RV, Kennington JL (Eds), *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*. Kluwer Academic Publishers, p. 1-75.

[10] **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 seqüenciamento em uma

máquina com penalidades por antecipação e atraso da produção, *In* Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional, SBPO, Fortaleza, p. 1649–1660.

[11] **Gupta, S. R. e Smith, J. S.** (2006), Algorithms for single machine total tardiness scheduling with sequence dependent setups, *European Journal of Operational Research*, v. 175, p. 722–739.

[12] **Hallah, R. M.** (2007), Minimizing total earliness and tardiness on a single machine using a hybrid heuristic, *Computers and Operations Research*, v. 34, p. 3126–3142.

[13] **Hino, C. M.; Ronconi, D. P. e Mendes, A. B.** (2005), Minimizing Earliness and Tardiness Penalties in a Single-Machine Problem with a Common Due Date, *European Journal of Operational Research*, v. 160, p. 190–201.

[14] **Kim, S. C. e Bobrowski, P. M.** (1994), Impact of sequence dependent setup time on job shop scheduling performance, *International Journal of Production Research*, v. 32, n. 7, p. 1503-1520.

[15] **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 and Operations Research*, v. 22, 857–869.

[16] **Li, G.** (1997), Single Machine Earliness and Tardiness Scheduling, *European Journal of Operational Research*, v. 96, p. 546–558.

[17] **Liaw, C. F.** (1999), A Branch-and-Bound Algorithm for the Single Machine Earliness and Tardiness Scheduling Problem, *Computers and Operations Research*, v. 26, p. 679–693.

[18] **Mazzini, R. e Armentano, V. A.** (2001), A Heuristic for Single Machine Scheduling with Early and Tardy Costs, *European Journal of Operational Research*, v. 128, p. 129–146.

[19] **Mladenovic, N. e Hansen, P.** (1997), Variable Neighborhood Search, *Computers and Operations Research*, v. 24, p. 1097–1100.

[20] **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, p. 29-38.

[21] **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 and Operations Research*, v. 31, p. 1727–1751.

[22] **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. 271–281.

[23] **Ying, K. C.** (2008), Minimizing earliness–tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm, *Computers and Industrial Engineering*, to appear. doi:10.1016/j.cie.2008.01.008.