v.2, n.3, p. 228-243, dez.1995

META-HEURÍSTICA PARA PROGRAMAÇÃO DA PRODUÇÃO COM TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQÜÊNCIA

Hamilton Carlos Massaro Santos
Depto. de Eng. de Sistemas -
Faculdade de Eng. Elétrica - UNICAMP
C.P.: 6101, cep 13081-970, Campinas - SP,
Fone (0192) 39-7914
e-mail: hamilton@densis.fee.unicamp.br

Paulo Morelato França
Prof. Dr. do Depto. de Eng. de Sistemas -
Faculdade de Eng. Elétrica - UNICAMP
C.P.: 6101, cep 13081-970, Campinas - SP,
Fone (0192) 39-7914
e-mail: franca@densis.fee.unicamp.br

Resumo

Este trabalho considera o problema de programação da produção, em uma máquina, de um conjunto de ordens de produção que podem ser agrupadas em famílias, sendo que os tempos de preparação entre essas famílias são dependentes da seqüência em que são executadas. Propõe-se um procedimento aproximado, baseado na meta-heurística de Busca Tabu, para a resolução deste problema. A função objetivo considera uma ponderação envolvendo os custos de preparação de máquina, uma penalidade por atraso em relação à data de entrega das ordens e o custo de estoque. O desempenho do método proposto é avaliado, computacionalmente, frente a três diferentes situações. 1) análise empírica de desempenho da heurística, em função dos parâmetros do problema; 2) comparação entre a heurística e regras de despacho tradicionais EDD e SPT; 3) emprego da heurística para a resolução de um problema prático real.

Palavras-chave: Sistemas de Manufatura, Programação da Produção, Métodos Aproximados, Meta-Heurísticas, Busca Tabu.

1. Introdução

O Problema de Programação da Produção em uma Máquina (PPPM) considerado consiste em definir a seqüência de execução de **n** ordens a serem processadas em uma única máquina. Cada ordem **j** tem um tempo de processamento t_j e uma data de entrega d_j. Considera-se o estudo de problemas em que o tempo

de preparação de máquina p_{ij} entre duas ordens **i** e **j** varia em função de atributos relativos às ordens, sendo esses tempos dependentes da seqüência em que as ordens **i** e **j** são executadas.

Este tipo de problema de programação ocorre com freqüência em muitos processos de fabricação. Um exemplo clássico é o da produção de itens de cores distintas. A mudança para a produção de um lote de uma certa cor ocasiona um tempo (ou custo) de preparação da máquina que é função da cor (atributo) do item anteriormente processado (BAKER, 1974).

Os problemas de programação envolvem, em geral, a perseguição de objetivos múltiplos e freqüentemente conflitantes (FRENCH, 1982). Os principais são: cumprir com as datas de entrega prometidas, minimizar a quantidade de ordens entregues após essas datas, minimizar o período total de programação, minimizar o tempo total em que as máquinas ficam ociosas e minimizar o custo de inventário.

Tais objetivos estão associados a custos de produção e podem ser expressos em função deles. Alguns desses custos variam com a programação, fazendo com que ela possa ser otimizada. Os custos mais relevantes são:

Custo de preparação de máquina;

Custo de ociosidade de máquina;

Custo de estoques;

Custo, ou penalidade, por atrasos.

Na realidade, o custo total associado a um programa de produção é uma composição desses custos. Dificilmente a programação é plenamente satisfatória com a consideração de um único desses componentes. Este trabalho se propõe a focalizar o problema mediante desenvolvimento e a análise empírica de desempenho de um método que leve em consideração uma ponderação de vários desses custos relevantes. Uma ferramenta computacional com tal foco tem a utilidade de confrontar soluções nas quais se pode dar diferenciada importância aos critérios e permitir tomar decisões dentro de um universo mais abrangente de considerações.

Nesta pesquisa, o critério adotado para avaliar um programa proposto é uma combinação do custo de preparação de máquina, do custo de estoque e uma penalidade por atraso. O efeito econômico provocado pela ociosidade de máquina não será considerado, devendo sua incorporação ser objeto de um futuro trabalho.

Os tempos e custos de preparação de máquina são gerados por ajustes e limpeza da máquina, necessários entre uma ordem e outra. Contudo, podem existir ordens que tenham as mesmas características de processamento e, quando programadas uma em seguida da outra, não geram nem tempo e nem custo de preparação. Tais ordens podem ser agrupadas em uma mesma família de preparação. Segundo MANSON & ANDERSON (1991) e WOODRUFF & SPEARMAN (1992), o agrupamento de ordens em famílias é crucial para se reduzirem tempos e custos de preparação de máquina.

A defasagem entre o instante de conclusão c_i e a data de entrega d_i de uma ordem **i** pode ocasionar tanto custo por estoque de produto acabado ($d_i > c_i$), quanto custo por atraso ($c_i > d_i$). Até há pouco tempo, a maioria dos trabalhos em programação considerava apenas os custos por atraso, mediante proibição ou imposição de penalidades. SEN & GUPTA (1984) apresentam um *survey* com muitas referências a esse enfoque.

Recentemente tem sido intensificada a consideração dos custos de estoque, junto aos demais custos que afetam a programação da produção. Segundo BAKER & SCUDDER (1990) isto se deve à adoção do conceito *just-in-time*, o qual prega que as ordens devem ser completadas o mais próximo possível das suas datas de entrega, minimizando estoques.

No trabalho de WOODRUFF & SPEARMAN (1992) é considerado um problema de programação da produção com agrupamento de ordens em famílias e com tempos de preparação dependentes da seqüência. Além disso, distinguem-se ali

duas classes de ordens: as ordens obrigatórias, que devem ser atendidas com prioridade, e as outras ordens, que serão atendidas caso caibam dentro do horizonte finito da programação. Os atrasos são proibidos, em vez de serem penalizados, como no presente trabalho.

Um enfoque semelhante ao adotado nesta pesquisa pode ser encontrado em LIMA (1993), que usa a teoria de conjuntos nebulosos para gerar as seqüências de ordens.

1.1 Modelagem do Problema

O PPPM com tempos de preparação dependentes da seqüência pode ser visto como um Problema de Caixeiro Viajante (PCV), no qual a execução de uma ordem **j** corresponde à visita ao cliente **j**, e p_{ij} ao tempo de viagem para ir do cliente **i** ao cliente **j**. Uma formalização dessa equivalência pode ser encontrada em BAKER (1974, pg. 94). Deve-se considerar uma *ordem fantasma* **0** que corresponde a um estado inicial e final de ajuste da máquina, que no PCV corresponde à cidade de origem e chegada do caixeiro. Assim, o problema tem n+1 ordens. Uma função de desempenho econômico para um programa *P*, $f(P)$, que considere custos por atraso, estoque e preparação de máquina pode ser definida como

$$f(P) = \sum_{k=1}^{n+1} \left[\gamma_{j_{k-1} j_k} p_{j_{k-1} j_k} + \alpha_{j_k} \left(c_{j_k} - d_{j_k} \right)^+ + \beta_{j_k} \left(d_{j_k} - c_{j_k} \right)^+ \right] \qquad (1.1)$$

onde, $\gamma_{j_{k-1} j_k}$ é o custo de preparação de máquina entre uma ordem na posição (**k-1**) do programa e a ordem seguinte a essa, na posição **k**. Esse custo é expresso em unidades monetárias (UM), por unidade de tempo. Os parâmetros α_{j_k} e β_{j_k} podem ser vistos como custos (ou penalidades) por atraso e estoque da ordem **j** na posição **k**, respectivamente. Também são expressos em UM / unidade de tempo. O símbolo $(x)^+$ significa que deve-se tomar o valor x, se x 0 e zero, caso contrário.

A expressão 1.1 está considerando custos diferentes para cada ordem **j**. Essas diferenças podem ocorrer em função de atributos específicos dessas ordens (cuidados na estocagem, multas contratuais por atraso, por exemplo) e em função do tamanho delas.

Admite-se neste trabalho que apenas os tamanhos das ordens influenciam os seus custos e, obviamente, que eles não influenciam os custos de preparação de máquina. Portanto, a expressão 1.1 pode ser reformulada da seguinte maneira:

$$f(P) = \sum_{k=1}^{n+1} \left[\alpha_1 p_{j_{k-1} j_k} + \alpha_2 \left(c_{j_k} - d_{j_k} \right)^+ q_{j_k} + \alpha_3 \left(d_{j_k} - c_{j_k} \right)^+ q_{j_k} \right] \qquad (1.2)$$

onde, q_{j_k} é a quantidade da ordem **j** na posição **k**, $_1$ é o custo de preparação de máquina em UM/tempo e $_2$ e $_3$ são os custos por atraso e por estoque, respectivamente, em (UM/tempo) x quantidade.

É importante observar que em problemas de programação nas quais a medida de desempenho não é regular, como é o caso de $f(P)$, o emprego de tempos de ociosidade como variáveis pode resultar em soluções com menores defasagens entre os instantes de conclusão e as datas de entrega de algumas ordens, diminuindo-se, assim, os custos com estoque de produtos acabados.

2. Métodos de Resolução

Os problemas de programação da produção têm recebido grande atenção por parte de pesquisadores, devido à sua extensa aplicação a problemas práticos. Além disso, a dificuldade inerente à sua resolução tem representado um desafio para a busca de métodos eficientes.

Devido à correlação do PPPM com tempos de preparação dependentes da seqüência com o PCV, muitos dos métodos desenvolvidos para a resolução do PCV podem ser adaptados para a resolução do PPPM. Com isso, a área de programação muito se beneficia, pois o PCV tem sido um dos problemas de Otimização Combinatória que mais contribuições têm recebido nas últimas décadas. Uma descrição de métodos exatos e aproximados já propostos pode ser encontrada em LAWLER *et al* (1985), LAPORTE (1992), SOLOMON & DESROSIERS (1988).

Estudos em complexidade têm classificado a maioria dos problemas de programação como "fáceis" ou "muito difíceis" (BLAZEWICZ *et al* 1991). O PPPM em questão encaixa-se no segundo caso.

Para tais problemas, os algoritmos exatos requerem um número de passos que cresce como uma função exponencial da dimensão do problema (clientes no caso do PCV, ordens no caso do PPPM).

Para resolver satisfatoriamente problemas de programação de interesse prático é imperioso lançar mão de técnicas de resolução ditas *aproximadas*, que embora sem a garantia de achar a solução ótima, são capazes de fornecer uma solução não muito distante dela, sem consumir grandes quantidades de tempo e memória computacionais. Esta é a razão de haver um grande esforço na direção de desenvolver heurísticas para problemas de programação. Focalizando o PCV, as principais técnicas heurísticas propostas para sua resolução enquadram-se em duas categorias:

Métodos de construção. Essas heurísticas são geralmente usadas para produzir uma rota no PCV. Elas são empregadas em problemas em que o objetivo é minimizar a distância total de viagem. No PPPM, as heurísticas de construção clássicas fornecem programas que buscam minimizar o tempo total de programação. Para o PPPM focalizado neste trabalho, cujo o objetivo é uma composição de custos por atraso, estoque e preparação de máquina, as heurísticas clássicas de construção devem sofrer uma adaptação para que boas soluções sejam obtidas. Em geral, porém, a qualidade das heurísticas de construção não é satisfatória, exigindo otimização adicional.

Métodos de melhoria. São métodos que aplicam mecanismos de perturbação em uma dada solução factível, com o objetivo de gerar uma vizinhança para prosseguir na busca. Eles também são conhecidos como métodos de busca local ou de vizinhança. A obtenção da solução factível de partida se dá por qualquer heurística de construção, ou mesmo por meio de uma seqüência aleatória das ordens. Aplicam-se a essa solução inicial mecanismos de perturbação, gerando outros programas vizinhos a ela. Seleciona-se, então, o programa com melhor valor na vizinhança gerada. Se for melhor que a solução anterior, ele passa a ser a nova solução. Os mecanismos de perturbação são aplicados sucessivamente, até que a melhor solução da vizinhança corrente não seja melhor que a solução que a gerou. Isso caracteriza um ótimo local e a busca termina.

O processo de busca, ao parar no primeiro ótimo local, pode estar deixando de avaliar outras regiões do espaço de soluções, algumas possivelmente contendo soluções melhores que o primeiro ótimo encontrado. Com a finalidade de não ficar restrito a ótimos locais, e ampliar o espaço de busca, desenvolveram-se métodos mais robustos e eficientes, capazes de transpor a otimalidade local. Tais métodos são chamados de *metaheurísticas*. Os mais conhecidos são os Algoritmos Genéticos, *Simulated Annealing* e a Busca Tabu (BT). Neste trabalho propomos técnicas de BT para solucionar o PPPM com tempos de preparação dependentes da seqüência.

3. Busca Tabu

Busca Tabu (GLOVER, 1989, 1990) e (GLOVER *et al*, 1993) é uma meta-heurística que se superpõe a uma heurística de busca local, com o objetivo de direcionar a busca para regiões com certas características desejáveis, como por exem-plo, regiões ainda não exploradas ou que pareçam muito promissoras. Além disso, BT tem o poder de transcender a otimalidade local, fazendo com que a busca visite muitos ótimos locais, talvez de melhor qualidade.

Ela usa o mesmo mecanismo de geração de vizinhança da heurística de busca local à qual está acoplada, e, a cada iteração, escolhe o melhor vizinho gerado para ser a nova solução corrente. Como já observado, a diferença para os métodos de busca local é que a BT, estando correntemente num ótimo local, permite escolher uma solução pior do que aquela que a gerou. Na realidade ela escolhe a solução "menos ruim" dentro da vizinhança. Porém, na geração da próxima vizinhança, o "novo melhor vizinho" provavelmente será o mínimo local já explorado, caracterizando assim uma ciclagem. Para prevenir ciclagens, deve-se estabelecer um mecanismo que proíba certos movimentos (tabus), de modo a evitar o retorno a soluções já exploradas.

Sendo as ciclagens causadas por movimentos inversos aos usados para atingir uma certa solução, é possível prevení-las proibindo a realização desses movimentos inversos por um certo número de iterações. É como se a busca ficasse dotada de um mecanismo de memória de curto prazo, para evitar seguir por caminhos já percorridos.

Tanto os movimentos quanto os seus reversos podem ser caracterizados por atributos a eles inerentes. Isso é interessante sob o ponto de vista computacional, pois é muito trabalhoso armazenar e lidar com todos os valores e estruturas que caracterizam uma solução. Assim, se um movimento que se deseja executar tiver atributos de um mo-vimento reverso anterior, que foi proibido, ele deve ser rejeitado, pois há uma grande chance de levar a soluções já visitadas.

Os atributos devem ser armazenados em estruturas especiais que são constantemente atualizadas: os atributos característicos de movimentos novos devem ser inseridos nela e os atributos já armazenados há algum tempo devem ser liberados, para que não impeçam a realização de movimentos que venham a explorar novas regiões. Por outro lado, a duração da permanência não deve ser muito curta, pois isso pode levar a ciclagens. Portanto, a duração da proibição é de importância fundamental para evitar ciclagens e direcionar convenientemente a busca.

Como os procedimentos com BT não terminam num ótimo local ("busca sem fim"), há a necessidade de se definir um critério de parada para o processo de busca, de forma a garantir um equilíbrio entre a obtenção de uma boa solução e o custo computacional para obtê-la.

4. Heurística Proposta

A heurística desenvolvida para a resolução do PPPM com tempos de preparação dependentes da seqüência é uma composição de três etapas:

Obtenção da solução inicial - Construção;

Melhoria da solução - Busca Local;

Busca por novas soluções - Busca Tabu.

A obtenção da solução inicial é conseguida usando-se uma adaptação da heurística de inserção de nós apresentada por SOLOMON (1987) para o contexto do Problema de Roteamento de Veículos (PRV). A etapa de melhoria da solução inicial é realizada por um procedimento de

busca local utilizando a heurística *2-opt* proposta inicialmente por LIN (1965) para solucionar o Problema do Caixeiro Viajante. A etapa de busca por novas soluções será realizada pela meta-heurística de Busca Tabu, em que o mecanismo de perturbação usado para a geração de novas soluções também é o *2-opt*. Uma aplicação deste enfoque para o PRV é relatada em PUREZA & FRANÇA (1991).

4.1 Considerações Iniciais

Antes da apresentação da heurística, há a necessidade de considerar alguns aspectos práticos e hipóteses adotadas para o PPPM considerado.

Todas as ordens estarão disponíveis para processamento no instante inicial da programação, chamado instante zero.

Não se consideram tempos de ociosidade de máquina como variáveis de decisão. Essa consideração teria implicações outras, além de uma possível diminuição do custo de estoque, tais como o aumento do *makespan*, aumento do custo de mão-de-obra parada e do custo de capital fixo.

Se uma operação já foi iniciada, ela deve ser completada antes que outro processamento seja iniciado na máquina, ou seja, não se admite preempção.

A máquina não pode processar mais de uma ordem ao mesmo tempo.

A máquina está sempre disponível; não se considera manutenção ou previsão de quebra.

4.2 Obtenção da Solução Inicial

Como a medida de desempenho adotada é uma ponderação de custos que são função da data de entrega (atraso de ordens, estoque de produtos) e de custos que são função exclusivamente da seqüência das ordens (preparação de máquina), optou-se por fazer uma adaptação para o PCV da heurística de inserção proposta por SOLOMON (1987). Ela foi desenvolvida para um PRV com janelas de tempo, onde estas considerações temporais também estão presentes. Essa heurística apresenta um bom compromisso entre qualidade de solução e tempo computacional.

Em termos do PCV, o primeiro passo da heurística é a escolha de um cliente para iniciar a rota. A seguir, o método utiliza dois critérios para a inserção de um novo cliente. O critério C_1(i, u, j) estabelece o melhor lugar para se inserir um cliente **u** ainda não alocado, entre os clientes adjacentes **i** e **j**, já alocados. O critério C_2(i, u, j) define qual o melhor cliente **u** a ser inserido no lugar definido por C_1(i, u, j). SOLOMON apresenta três enfoques para o emprego desses critérios, os quais diferem na avaliação do local e do cliente a ser inserido na rota em construção.

O terceiro enfoque proposto por SOLOMON utiliza a ponderação de três sub-critérios para avaliação do lugar de inserção e do cliente a ser inserido. Esses sub-critérios possuem uma proximidade com a medida de desempenho adotada para o PPPM.

O primeiro sub-critério utilizado por Solomon é

$$C_{11}(i,u,j) = d_{iu} + d_{uj} - \mu\, d_{ij} \qquad (4.1)$$

sendo d_{ij} a distância entre o cliente **i** e o cliente **j.** O parâmetro confere flexibilidade ao critério. Adotando-se = 1, ele avalia o acréscimo da distância na rota parcial, caso o cliente **u** seja inserido entre dois clientes adjacentes, já inseridos na rota, **i** e **j**.

Se, ao invés das distâncias, fossem adotados os tempos de viagem entre os clientes, esse sub-critério estaria avaliando o acréscimo de tempo, na rota, pela inserção de **u** entre **i** e **j**. Com a correlação existente entre o tempo de viagem do PCV e o tempo de preparação de máquina no PPPM, pode-

se usar esse último na expressão 4.1, para que esse sub-critério avalie os custos com preparação de máquina na heurística de construção do programa inicial. Desta forma tem-se:

$$C_{11}(i,u,j) = p_{iu} + p_{uj} - p_{ij} \quad (4.2)$$

O segundo sub-critério de SOLOMON é

$$C_{12}(i,u,j) = b_{uj} - b_j \quad (4.3)$$

com b_{uj} sendo o novo instante para início de atendimento de **j**, dado que **u** foi inserido antes dele, e b_j o instante de início de **j,** antes da inserção de **u**. Esse sub-critério pondera o acréscimo de tempo na rota parcial devido à inserção de **u** entre **i** e **j.**

Não se está considerando, de forma direta, a ponderação dos custos por instantes de conclusão. Contudo, sabe-se que o acréscimo de tempo em um programa (ou rota) altera as defasagens entre os instantes de conclusão e as datas de entrega, a partir do ponto em que se deu esse acréscimo. Um acréscimo de tempo aumenta os atrasos já existentes e diminui os tempos de espera, ou os transforma em atrasos. Desta forma, o segundo sub-critério deve ser alterado para penalizar os custos de atraso no processo de construção do programa. Porém, apenas o atraso relativo à ordem **u** será considerado, o que confere um caráter "míope" ao sub-critério.

Portanto, o segundo sub-critério adotado foi reformulado para

$$C_{12}(i,u,j) = \left(\left(b_u + t_u\right) - d_u\right)^+ \quad (4.4)$$

O terceiro sub-critério, como sugerido por SOLOMON, é representado pela expressão

$$C_{13}(i,u,j) = l_u - b_u \quad (4.5)$$

onde l_u é o instante de início de atendimento do cliente **u**. Esse sub-critério reflete o tempo de espera para se atender o cliente **u**, caso ele seja inserido entre os clientes **i** e **j**. A expressão que penaliza a espera de uma ordem **u**, se essa for inserida entre duas ordens adjacentes **i** e **j** será:

$$C_{13}(i,u,j) = \left(d_u - \left(b_u + t_u\right)\right)^+ \quad (4.6)$$

Deve-se observar que os sub-critérios C_{12}(i, u, j) e C_{13}(i, u, j) estão analisando o aumento do tempo de atraso e o aumento do tempo de espera, respectivamente, apenas na ordem **u** a ser inserida e não no total do programa parcial em construção. Devido a este caráter "míope" deve-se esperar uma baixa qualidade da solução inicial a ser obtida.

SOLOMON considera o critério para a definição do cliente a ser inserido igual ao critério para definição do local de inserção, o que também é adotado aqui. Logo, para toda ordem não programada **u**, encontram-se todos os valores do critério C_1 possíveis. O menor valor de C_1 define a ordem **u** a ser inserida e o local (ordens **i** e **j**) de sua inserção. Assim,

$$C_1(i,u,j) = Min\left(\alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j) + \alpha_3 C_{13}(i,u,j)\right) \quad (4.7)$$

Como se está considerando que os custos por atraso e estoque dependem não só dos tempos de atraso e espera, mas também das quantidades de produtos das ordens atrasadas ou em estoque, a expressão 4.7 deve ser reformulada para a consideração dessas quantidades. Assim,

$$C_1(i,u,j) = Min\left(\alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j) q_u + \alpha_3 C_{13}(i,u,j) q_u\right) \quad (4.8)$$

4.3 Melhoria da Solução - Busca Local

A partir da solução inicial, aplica-se um método de melhoria baseado na heurística *2-opt* (LIN, 1965). Essa heurística gera a vizinhança de uma solução, removendo um par de arestas não adjacentes e reconectando as duas cadeias resultantes de forma a constituir uma nova solução. Portanto, em um programa de n+1 ordens o método *2-opt*

gera uma vizinhança de tamanho $\binom{n+1}{2} - [(n+1) - 1]$.

Em cada movimento do *2-opt* ocorrem mudanças nos instantes de início e término das ordens, a partir da ordem que antecede a primeira aresta eliminada. Essas mudanças resultam na alteração de tempos de espera e atraso. Como algumas ordens mudam de lugar no programa, tem-se também a possibilidade de alterações de alguns tempos de preparação.

A figura 1 ilustra as alterações, decorrentes de um movimento realizado pelo método de melhoria *2-opt*, em um programa com cinco ordens. Nessa figura deve-se observar a seguinte notação:

A_j = atraso da ordem j
E_j = tempo de espera da ordem j
d_j = data de entrega da ordem j
p_{ij} = tempo de preparação entre ordens i e j

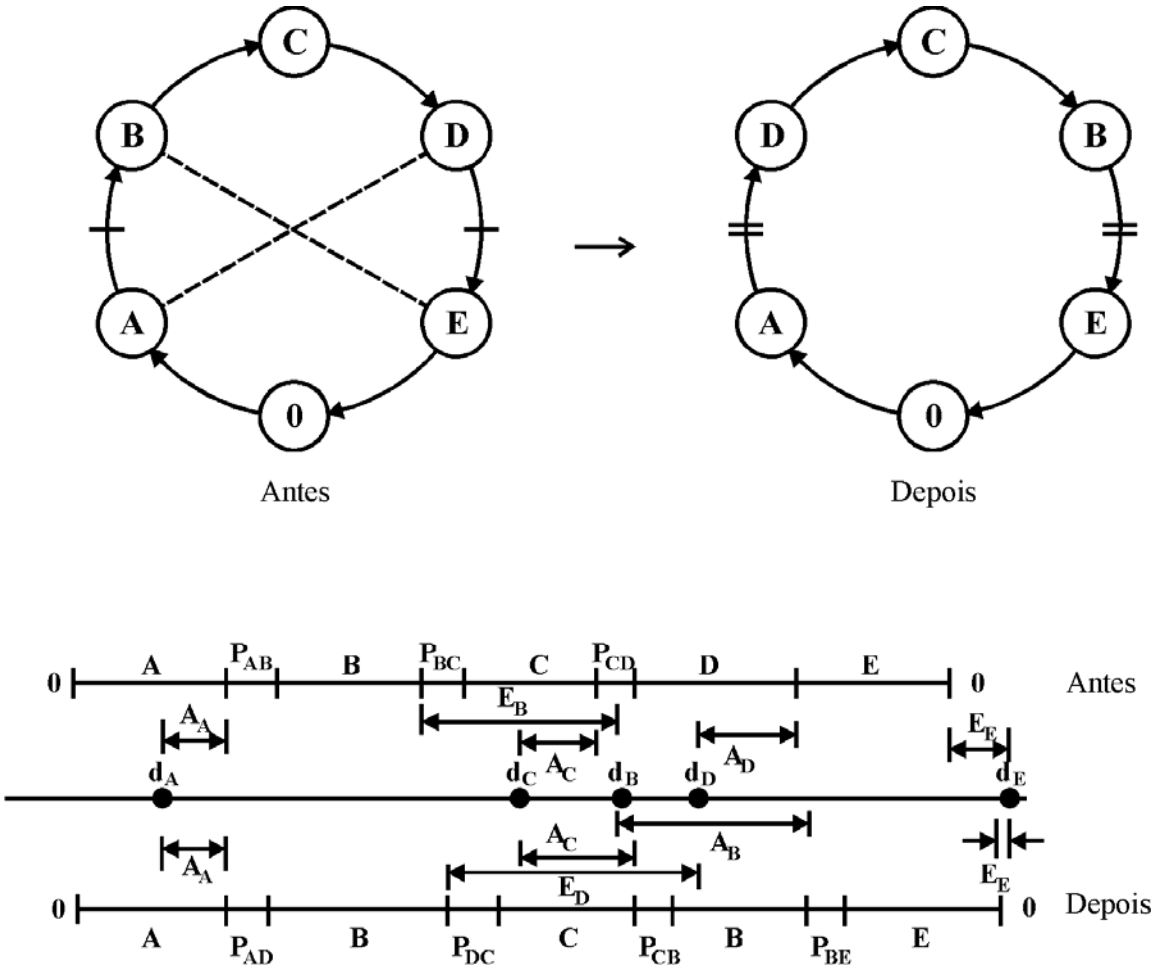

Figura 1 : Alterações Decorrentes de um Movimento *2-opt*

A solução final encontrada pela heurística *2-opt* é chamada de primeiro ótimo local. Adotou-se a expressão (1.2) como medida de desempenho para avaliação das soluções geradas pelo *2-opt*.

4.4 Busca de Novas Soluções - Busca Tabu

A partir do primeiro ótimo local, aplica-se BT com a finalidade de explorar melho-

res soluções.

No procedimento desenvolvido, os atributos escolhidos para representar movimentos foram as *arestas eliminadas* e as *arestas adicionadas* envolvidas. Como estruturas para o armazenamento dessas arestas foram construídas duas matrizes, uma para as arestas eliminadas e outra para as adicionadas. Uma célula [ij] da matriz corresponde a uma aresta do programa, em que **i** é a ordem antecessora imediata da ordem **j**. O valor da célula é o número de iterações em que cada atributo será considerado proibido (tabu). Ele é obtido pela adição da iteração atual, com o número de iterações em que o atributo deve ficar ativo. Este é um parâmetro, que chamaremos de ***tabu tag***, que pode ser fixo ou variável. O uso dele como variável tem-se mostrado mais vantajoso para evitar ciclagens, além de fornecer melhores soluções na busca (TAILLARD, 1990). Na realidade, os melhores resultados são conseguidos mediante a geração de valores aleatórios para cada *tabu tag*.

Note que quanto maiores os valores de *tabu tag*, mais tempo as arestas permanecem ativas nas matrizes, contribuindo para a proibição de movimentos, o que torna a busca mais restritiva; e vice-versa.

Para cada nova solução gerada, a heurística consulta essas duas matrizes. Verifica se as arestas adicionadas no movimento coincidem com arestas eliminadas em movimentos recentes (ativas na matriz de arestas eliminadas) e se as arestas eliminadas neste movimento coincidem com arestas adicionadas em movimentos recentes. Se a soma do número de coincidências for maior que um parâmetro, **maxted,** previamente definido, esse movimento é considerado tabu, ou seja, não é realizado. O valor de maxted pode variar de 1 a 4, sendo que quanto menor o seu valor, mais restritiva será a busca. A busca termina quando o número de iterações chegar a um valor préfixado, **limit**.

5. Resultados Computacionais

A heurística foi implementada em linguagem Pascal e os testes computacionais foram realizados em uma estação de trabalho Sun modelo *SPARCclassic*.

A fim de avaliar o desempenho empírico da heurística proposta, dois conjuntos de testes foram realizados. No primeiro, buscou-se calibrar os parâmetros que influenciam o desempenho da BT, como **maxted** e ***tabu tag,*** que determinam o adequado compromisso entre o quão restritivo ou complacente deve ser o processo de busca. No segundo, é mostrado o desempenho da heurística sobre um conjunto de problemas gerados aleatoriamente. Para isso, uma primeira bateria de testes mostra as melhorias alcançadas em cada uma das três fases da heurística: construtiva, busca local e busca tabu. Uma segunda bateria compara a heurística com as regras de despacho EDD e SPT, usadas para programar a produção de sistemas com datas de entrega. Para finalizar os testes comparativos, a heurística proposta é usada para resolver um problema real de programação encontrado em uma indústria de fabricação de chapas plásticas.

Na geração dos conjuntos de dados, cada ordem teve o seu tempo de processamento gerado aleatoriamente a partir de uma distribuição uniforme. Foram usados dois diferentes intervalos de geração, [5-55] e [10-80]. As datas de entrega foram determinadas da mesma forma, com os intervalos de geração sendo delimitados por uma data de entrega mínima, chamada de **datamin**, e pelo horizonte de programação **h**. Foram usadas várias combinações diferentes de **datamin** e **h**.

A distribuição das ordens pelas **nf** famílias de ajustes da máquina também foi feita de forma aleatória e uniforme. Os tempos de preparação entre essas famílias foram gerados de forma proporcional à diferença entre elas, ou seja, o tempo de preparação

entre as famílias **i** e **j**, é calculado pela expressão

$$i - j \ * tprep * (\pm 0,05)$$

onde **tprep** é um multiplicador. O fator 0,05 introduz maior aleatoriedade aos tempos gerados, para impedir que haja muitas ordens com o mesmo valor de tempo de preparação. Tipicamente, os valores usados para **tprep** situam-se numa faixa entre 3 e 10.

A escolha da primeira ordem da seqüência em construção foi feita pelo critério da ordem com a menor data de entrega, que tem o efeito prático de minimizar as defasagens entre os instantes de conclusão e as datas de entrega.

Um aspecto importante no emprego da heurística é o tempo de execução na resolução de problemas com diferentes tamanhos. Sabe-se que tal tempo de execução é exponencial, pois é proporcional ao tamanho da vizinhança gerada a cada passo da busca. Apenas para ilustrar, apurou-se o tempo de execução para a resolução de quatro problemas de tamanhos diferentes, usando um valor de **limit** de 500 iterações. O resultado está mostrado na Tabela 1.

Tabela 1 - Tempos de Execução em Função do Tamanho do Problema(n)

n	tempo (s)
20	27
50	338
75	1117
100	2769

No primeiro conjunto de testes foram determinados valores convenientes para os parâmetros que influenciam o processo de busca, como maxted, intervalo de geração dos tabu tags e limit. Para isso foram usados apenas dois problemas de 20 e 50 ordens, gerados aleatoriamente de acordo com os intervalos descritos acima. Observou-se que, em média, a heurística tem melhor desempenho com valores de maxted iguais a 2 ou 3. Quanto aos intervalos de tabu-tag, verificou-se que um bom equilíbrio entre qualidade de solução e risco de ciclagens se dá para valores gerados aleatoriamente entre 25 e 40.

5.1 Análise do Desempenho da Heurística

Com os parâmentros de BT calibrados pelas experiências relatadas acima, faz-se necessário avaliar o comportamento global da heurística frente a variações dos diversos parâmetros próprios do PPPM. Para este segundo conjunto de testes, mostramos apenas as variações relativas aos parâmetros tempo de preparação (tprep) e data mínima de entrega (datamin). Análises quanto à variação de outros parâmetros, como número de famílias (nf), critério para escolha da tarefa de inicialização do programa, etc. podem ser encontradas em SANTOS (1994).

Definimos duas classes de valores para tprep e datamin: baixo e alto. Com isso se definem quatro casos possíveis:

	tprep	datamin

caso 1	baixo	baixo
caso 2	baixo	alto
caso 3	alto	baixo
caso 4	alto	alto

Para cada caso é analisada a variação dos três parâmetros de custo usados na função objetivo do problema: custo de preparação ($_1$), custo de atraso ($_2$) e custo de adiantamento ($_3$). Para cada possível combinação desses parâmetros, foram gerados 10 diferentes instâncias por meio de diferentes sementes. O problema escolhido para os testes tem 50 ordens. Os resultados são apresentados na Tabela 2.

Tabela 2 - Análise da Heurística para Diferentes Problemas

Caso	Comp dos Custos			Tempo	Tempo	Custo	Custo	Custo	Custo	% de Melhoria		
	$_1$	$_2$	$_3$	Atraso	Espera	Prepar.	Atraso	Estoque	Total	Sol-OL	OL-BT	Sol-BT
1	0	0	1	4489	2380	0	0	76563	76563	32,2	4,2	35,0
	0	1	0	2046	5845	0	76173	0	76173	83,8	14,0	86,1
	1	0	0	18657	22903	12	0	0	12	0,0	0,0	0,0
	40	0,1	0,1	2255	3840	7753	9199	12724	29676	17,9	2,1	19,6
2	0	0	1	2021	14859	0	0	416425	416425	20,9	1,7	22,2
	0	1	0	360	16419	0	11152	0	11152	96,2	11,4	96,7
	1	0	0	11765	28013	12	0	0	12	0,0	0,0	0,0
	40	0,1	0,1	657	15695	10068	3554	45796	59418	18,9	2,0	20,6
3	0	0	1	19883	142	0	0	5789	5789	9,5	1,1	10,5
	0	1	0	6489	5724	0	220825	0	220825	77,3	9,4	79,4
	1	0	0	19132	22672	40	0	0	40	0,0	0,0	0,0
	40	0,1	0,1	4099	3874	12023	18763	12260	43046	17,0	4,1	20,4
4	0	0	1	14566	5743	0	0	145068	145068	32,4	0,0	32,4
	0	1	0	2447	15231	0	64380	0	64380	91,0	9,8	91,9
	1	0	0	12174	27716	40	0	0	40	0,0	0,0	0,0
	40	0,1	0,1	2323	12244	15616	12853	36264	64733	15,2	2,6	17,4

Os valores representam a média calculada sobre as 10 instâncias. As três colunas de porcentagem de melhoria referem-se, respectivamente, à melhoria alcançada pela fase de busca local, ou seja, entre a solução inicial e o primeiro ótimo local, à melhoria da fase tabu e à melhoria total alcançada.

Pode-se observar, pelas percentagens de melhoria entre as três etapas da heurística, que ela promove uma grande melhoria até o primeiro ótimo local, indicando que a heurística de construção da solução inicial não é muito eficiente. Essa pouca eficiência na construção da solução de partida está no fato de se empregar uma adaptação da heurística de SOLOMON que considera, para cada inserção, apenas uma análise local (míope), ponderando somente os custos das ordens a serem inseridas. Se em cada inserção fosse feita uma ponderação dos custos de atraso, estoque e preparação, em todo o sub-programa e não apenas nela mesma, a solução de partida, embora mais custosa e demorada, poderia ser de melhor

qualidade. Porém, devido às características da meta-heurística BT, isso afetaria pouco a solução final alcançada.

Para todos os casos, quando se tem composições de custo que ponderam apenas o custo de estoque ou o custo de atraso ($_1$=0, $_2$=0, $_3$=1 e $_1$=0, $_2$=1, $_3$=0, respectivamente), a heurística emprega um grande esforço para a obtenção da solução, em especial até o primeiro ótimo local. Isso ocorre porque não há objetivos antagônicos nessas composições de custo. Assim, ao se ponderar apenas o custo de estoque, os movimentos com o *2-opt* que criam ou aumentam os tempos de preparação, irão resultar no aumento dos tempos de atraso e diminuição dos tempos de espera, ou a transformação desses em tempos de atraso. Como se está objetivando apenas a diminuição dos custos de estoque, tal movimento reduz muito tal custo, pois não tem os aumentos inversos de custos de atraso e de preparação de máquina. O mesmo raciocínio se aplica à composição de custos que pondera apenas o custo de atraso.

Para a composição de custos que pondera apenas o custo de preparação de máquina ($_1$=1, $_2$=0, $_3$=0), sempre ocorre não haver melhoria após a solução de partida. Isto se deve ao fato de que os tempos de preparação entre famílias são gerados de forma proporcional à diferença entre essas famílias. É simples verificar que nesses casos, o problema é resolvido de forma ótima pela heurística de inserção.

5.2 Heurística Frente a Regras de Despacho Tradicionais

É muito comum o emprego de regras de despacho para a resolução de problemas de programação. Uma das principais características dessas regras é a rapidez na obtenção dos programas. Duas das principais regras de despacho são: SPT (*Shortest Processing Time* - Menor Tempo de Processamento) e EDD (*Earliest Due Date* - Menor Data de Entrega). A regra SPT consiste basicamente em se organizar as ordens em uma seqüência crescente de tempos de processamento. A regra EDD consiste na programação das ordens em uma seqüência crescente das datas de entrega.

Para se verificar a eficiência da heurística desenvolvida frente a tais regras, resolveu-se um mesmo PPPM, com tempos de processamento dependentes da seqüência, usando os três procedimentos. Os resultados estão apresentados na Tabela 3. Os custos mostrados foram todos calculados da mesma forma, por meio da expressão (1.2).

Tabela 3 - Regras de Despacho vs. Heurística

	Custo				Tempo	
	Prepar.	Atraso	Estoque	Total	Atraso	Espera
EDD	11400,9	36809,7	8,3	48218,9	84310,5	24,3
SPT	10015,8	48363,9	1781,0	60160,7	78321,9	11983,4
Heurística	846,5	1380,1	1068,3	3294,9	2397,2	4296,4

Os resultados falam por si: as regras de despacho EDD e SPT não são adequadas para resolução de PPPM com tempos de preparação dependentes da seqüência. Devido ao fato da heurística ter sido desenvolvida considerando-se tais tempos, ela apresenta uma eficiência muitíssimo superior à de tais regras, largamente utilizadas em ambientes de programação da produção.

5.3 Análise da Heurística em uma Situação Prática Real

A heurística já foi analisada em problemas com dados gerados de forma aleatória e frente a regras de despacho. Agora ela é utilizada em um problema com dados reais enfrentado por uma empresa que fabrica chapas poliméricas em máquinas de extrusão. As chapas a serem produzidas podem diferir umas das outras basicamente pela cor, espessura, comprimento e largura. O tempo de preparação entre duas ordens será o maior dos tempos para ajuste de um desses fatores, já que tais ajustes podem ocorrer simultaneamente. Não há aqui agrupamento em famílias.

A matriz de tempos de preparação não é simétrica, ou seja, para se ajustar a máquina entre uma ordem **i** e uma ordem **j**, gasta-se um tempo diferente do tempo entre a ordem **j** e a ordem **i**. Esta característica é devida principalmente à cor.

A heurística foi aplicada para a resolução de dois problemas da empresa, com ordens a serem programadas em duas máquinas para um certo mês.

Os resultados obtidos encontram-se na Tabela 4. Nas resoluções não foram usados os custos reais de preparação de máquina, atraso e estoque, já que não se teve acesso a eles. Foram utilizadas, entretanto, duas combinações de custos para cada caso, para verificar a robustez da heurística face a diferentes cenários.

Tabela 4 - Heurística em um Problema Real de Programação

Máquina	N de Ordens	Custo Prepar.	Custo Atraso	Custo Estoque	Custo Total	% de Melhoria		
						Sol-OL	OL-BT	SOL-BT
$_1 = 4, \quad _2 = 0{,}01, \quad _3 = 0{,}01$								
1	42	1920	119428	14517	14517	59,1	25,2	69,4
3	115	5960	906301	28154	940415	63,8	1,4	64,3
$_1 = 2, \quad _2 = 0{,}01, \quad _3 = 1$								
1	42	1130	159122	20768	181020	34,6	17,9	46,3
3	115	3890	1212271	832	1216993	51,4	2,1	52,4

Verifica-se que o comportamento adotado pela heurística na resolução desse caso prático é similar ao empregado para os casos aleatórios, sendo que ela teve uma boa eficiência na programação da máquina 1, conseguindo, inclusive um bom desempenho na fase tabu. Para a máquina 3 não houve uma devida calibragem de parâmetros, o que pode ter resultado no fraco desempenho da fase tabu. Contudo, percebe-se que o método de melhoria *2-opt* obtém bons resultados para esse problema, já que ocorreu uma grande melhoria até a obtenção do primeiro ótimo local.

6. Conclusões Gerais

A principal contribuição deste trabalho está no desenvolvimento e análise empírica de desempenho de um método de resolução de problemas de programação da produção em uma máquina, considerando-se:

- tempos de preparação dependentes da seqüência;
- agrupamento em famílias de ordens com ajuste de máquina semelhantes;
- ponderação dos custos de atraso, estoque e preparação de máquina como medida

de desempenho dos programas de produção gerados;
- a possibilidade de análise das conseqüências de se interferir nas datas de entrega, possibilitando negociação com clientes.

Dos resultados obtidos, várias conclusões podem ser tiradas quanto ao uso da heurística e dos métodos empregados nas suas três diferentes fases: construção da solução factível inicial, busca local e Busca Tabu (BT). As principais são:
- Quanto menor o problema, mais relaxada deve ser a BT, pois do contrário não sobram vizinhos com boas carcterísticas para se prosseguir na busca;
- Tempos elevados de preparação de máquina, ou um número grande de famílias, aumentam os instantes de término das ordens, aumentando, assim, os tempos e custos de atraso e diminuindo os tempos de adiantamento para entrega e os custos com estoque. Nesse caso, deve-se evitar datas de entrega próximas ao início do horizonte de programação. Caso os tempos de preparação de máquinas ou o número de famílias sejam pequenos, deve-se evitar ordens com datas de entrega concentradas no final do horizonte de planejamento;
- A heurística é aplicável a problemas cuja medida de desempenho objetive apenas minimizar custos com atraso ou custos com estoque;
- Comparativamente à heurística proposta, as regras de despacho EDD e SPT mostraram-se muito pobres na resolução do PPPM com tempos de preparação dependentes da seqüência;
- A heurística obteve bons resultados na resolução de um problema prático com tempos de preparação de máquina, tempos de processamento e datas de entrega reais.
- O modelo de programação adotado é bem adequado à realidade e pode ser utilizado para programar a produção de forma mais efetiva, aumentando a eficiência produtiva, com reduções nos custos de estoque e preparação de máquina, além de um melhor cumprimento das datas de entrega.

Agradecimentos

Os autores agradecem Vitória M. Pureza pela cessão de parte do código que foi utilizado no trabalho. Esta pesquisa foi parcialmente apoiada pelo CNPq mediante bolsas de mestrado e pesquisa.

Referências Bibliográficas:

BAKER, K. R.: *Introduction to Sequencing and Scheduling.* John Wiley & Sons, New York, 1974.

BAKER, K. R. & SCUDDER, G. D.: "Sequencing with Earliness and Tardiness Penalties: A Review". *Operations Research.* Vol.38, N 1, p. 22-36, 1990.

BLAZEWICZ, J.; DROR, M. & WEGLARZ, J.: "Mathematical Programming Formulations for Machine Scheduling: A Survey". *European Journal of Operations Research.* Vol. 51, p. 283-300, 1991.

FRENCH, S.: *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop.* Ellis Horwood, 1982.

GLOVER, F.: "Tabu Search, Part I". *ORSA Journal on Computing.* Vol. 1, N 3, p. 190-206, 1989.

GLOVER, F.: "Tabu Search, Part II". *ORSA Journal on Computing.* Vol. 2, N 1, p. 4-32, 1990.

GLOVER, F.; LAGUNA, M.; TAILLARD, E. & DE WERRA, D.: "Tabu Search". *Annals of Operations Research.* Vol. 41, 1993.

LAPORTE, G.: "The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms". *European Journal of Operations Research.* Vol. 59, p. 231-247, 1992.

LAWLER E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G. & SHMOYS, D. B. (ED.): *The Traveling Salesman Problem.* John Wiley & Sons, 1985.

LIMA, P. C.: *Um Sistema de Programação Finita Baseado em Lógica Nebulosa.* Tese de Doutorado. FEM-UNICAMP, 1993.

LIN, S.: "Computer Solutions of the Traveling Salesman Problem". *Bell Syst. Tech. J.*. Vol. 44, p. 2245, 1965.

MANSON, A. J. & ANDERSON, E. J.: "Minimizing Flow Time on a Single Machine with Job Classes and Setup Time". *Naval Research Logistics Quarterly.* Vol. 38, p. 333-350, 1991.

PUREZA V. M. & FRANÇA, P. M.: *Solving Vehicle Routing Problems via Tabu Search Metaheuristic*. Pub. # 747, Centre de Recherche sur les Transports. U. de Montréal, 1991.

SANTOS, H. C. M.: *Programação da Produção em Uma Máquina com Tempos de Preparação Dependentes da Seqüência e Penalidades*. Tese de Mestrado. FEE-UNICAMP, 1994.

SEN, T. & GUPTA, S. K.: "A State-of-Art Survey of Static Scheduling Research Involving Due Dates". *OMEGA.*. Vol. 12, N 1, p. 63-76, 1984.

SOLOMON, M. M.: "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". *Operations Research.* Vol. 35, p. 254-265, 1987.

SOLOMON, M. M. & DESROSIERS, J.: "Time Window Constrained Routing and Scheduling Problems". *Transportation Science.* Vol. 22, N 1, p. 1-13, 1988.

TAILLARD, E.: *Robust Tabu Search for the Quadratic Assignment Problem*. Département de Mathématiques, École Polytechnique Fédérale de Lausanne, Working Paper ORWP 90/10. Suiça, 1990.

WOODRUFF, D. L. & SPEARMAN, M. L.: "Sequencing and Batching for Two Classes of Jobs with Deadlines and Setup Times". *Production and Operations Management.* Vol. 1, N 1, p. 87-102, 1992.

METAHEURISTIC FOR SCHEDULING WITH DEPENDENT SETUP TIMES

Abstract

This article focuses on the one machine scheduling problem where jobs can be grouped in classes with the same machine setups. The setup times between classes are sequence dependent. An approximation method based on Tabu Search metaheuristic is proposed. The

objetive is to minimize the weighted sum of setup costs, tardiness and inventory holding costs. The performance of the heuristic is evaluated through three sets of computational tests: 1) empirical performance analysis of the heuristic with different data sets; 2) comparison between the heuristic and the well known dispatching rules EDD and SPT; 3) application of the heuristic for solving a real life scheduling problem.

Key-words: manufacturing systems, machine scheduling, approximation methods, metaheuristics, Tabu Search