Universidade Estadual de Campinas
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

UNICAMP

Programação da Produção em Uma Máquina com Tempos de Preparação Dependentes da Sequência e Penalidades

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de **Mestre em Engenharia Elétrica**.

por

Hamilton Carlos Massaro Santos
Engenheiro de Produção – Materiais, DEP/UFSCar

Campinas, 7 de dezembro de 1994

Banca Examinadora:

Paulo Morelato França	–	**Orientador**
Paulo Corrêa Lima	–	**FEM/UNICAMP**
Vinícius Amaral Armentano	–	**FEE/UNICAMP**

À

Marcela

aos meus pais

Milton e Claudete

e aos meus irmãos

Hânia e Haroldo

dedico

AGRADECIMENTOS

Agradeço a todos que me incentivaram e colaboraram para a realização deste trabalho e em especial

- Ao Prof. Paulo Morelato França pela orientação desta tese, estímulo à pesquisa e conselhos acadêmicos e pessoais

- Ao Prof. Vinícius Amaral Armentano pela atenção e interesse ao longo deste trabalho

- Ao Prof. Paulo Corrêa Lima pela atenção, colaborando com dados práticos e auxiliando na interpretação desses

- À Vitória M. M. Pureza pela participação, os valiosos *insights* em etapas deste trabalho e pela cessão de parte dos códigos utilizados na tese

- A José Rodrigues dos Santos Filho, Débora P. Ronconi, Cintia R. Scrich, Regina E. Berretta e Franklina M. B. de Toledo pelas valiosas discussões sobre este trabalho

- A Walcir , Cássio e Gelson pelo apoio computacional

- Aos Professores e funcionários do DENSIS pelo apoio

- Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico pelo apoio financeiro

RESUMO

Este trabalho considera o problema de programação da produção de uma máquina, onde as ordens de produção podem ser agrupadas em famílias com um mesmo ajuste de máquina e os tempos de preparação entre essas famílias são dependentes da sequência das ordens. Propõe-se uma heurística baseada na meta-heurística de Busca Tabu para a resolução deste problema. A função objetivo considera uma ponderação dos custos de preparação de máquina, atraso em relação à data de entrega e estoque. São apresentados vários resultados computacionais obtidos com a heurística. Esses resultados referem-se à: 1) Análise do comportamento do processo de busca com a aplicação da heurística; 2) Calibragem de parâmetros empregados pela heurística; 3) Análise do desempenho da heurística frente a diferentes variações de dados de problemas; 4) Análise de diferentes atributos utilizados pela busca tabu; 5) Comparação entre a heurística e regras de despaço tradicionais; 6) Emprego da heurística para a resolução de um problema prático real.

ABSTRACT

This work focuses on the one machine scheduling problem with sequence dependent setup times, the jobs can be grouped in classes with the same machine setups. The setup times between classes are sequence dependent. A heuristic based on Tabu Search is proposed. The objective is minimize the weighted sum of setup costs, tardiness and inventory. Computational tests are performed and theirs results are presented. This results are referring to: 1) Analysis of the search process behaviour; 2) Calibration of heuritic parameters; 3) Performance analysis of the heuristic with diferent data sets; 4) Analysis of different atributes considered in the heuristc; 5) Comparison between the heuristic and well known dispatching rules; 6) Application of the heuristic to solve a practical scheduling problem.

ÍNDICE

Página

Página

Capítulo 1

Introdução

Em diversos sistemas encontra-se o seguinte problema: várias atividades devem ser executadas em horizontes de tempo determinados, necessitando para tal empregar diferentes recursos. Uma atividade pode necessitar de vários recursos para a sua realização. Em geral alguns desses recursos servem a várias atividades e se um desses recursos estiver sendo usado em uma atividade, não estará disponível para qualquer outra.

Esses sistemas necessitam então passar por um processo de tomada-de-decisão, que determine a sequência de execução das atividades em cada recurso e os instantes em que se darão essas execuções, objetivando obter um bom ou ótimo valor de alguma medida de desempenho (por exemplo, o menor tempo para a realização dessas atividades).

Tais processos de tomada de decisão são denominados de **programação** e o resultado de um problema de programação é denominado **programa**[1], o qual deve ser seguido para se obter o(s) objetivo(s) almejado(s).

Muitos dos desenvolvimentos no campo da programação são motivados por problemas de manufatura (Baker 1974), sendo tratados como **problemas de programação de produção (PPP)**. Sendo assim, o emprego do vocabulário de manufatura passou a ser natural na descrição de problemas de programação. Desta forma, recursos geralmente são tratados como **máquinas** e atividades como **ordens de produção**.

Às vezes uma ordem de produção pode ser constituída de atividades elementares, cada qual necessitando de processos ou máquinas diferentes e que são inter-relacionadas por

---

[1]Esse termo não deve ser confundido com programa computacional, o qual trataremos por algoritmo.

restrições de precedência. Tais atividades elementares são definidas como **operações**.

Alguns profissionais dão ao termo programação uma conotação mais ampla que a enfocada aqui, sendo programação empregada com o sentido de planejamento.

Durante esse trabalho o termo **programação** é adotado para designar o processo de tomada-de-decisão que determine, para cada máquina, a sequência em que as ordens de produção devem ser processadas e o instante de processamento de cada uma dessas ordens, ou seja, o programa de produção. O termo **ordem** é adotado para designar ordem ou lote de produção ou simplesmente tarefa a ser programada. Isto por serem estes termos os mais empregados pelos profissionais envolvidos em sistemas de manufatura.

Os objetivos buscados em problemas de programação de produção são, em geral, numerosos, complexos e frequentemente conflitantes (French 1982). Dentre esses vários objetivos os principais são:

- Cumprir com as datas-de-entrega prometidas aos clientes, ou minimizar a quantidade de produtos entregues após essa. Caso contrário perde-se a confiança dos clientes, o que reflete em penalidades financeiras;
- Minimizar o período total da programação. Se todas as ordens de produção foram completadas, as máquinas estão disponíveis para outras ordens;
- Minimizar o tempo total no qual as máquinas ficam ociosas, já que máquina ociosa significa capital ocioso;
- Minimizar o custo de inventário, não apenas de produto acabado mas também de matéria-prima e material em processo.

Esses objetivos estão associados a custos de produção e podem ser expressos em função desses. Alguns desses custos variam com a programação da produção, logo, a alteração dessa pode reduzir tais custos, satisfazendo os objetivos expostos. Esses custos são:

- Custo de preparação de máquina. Quando da troca de ordem de produção na máquina, o processamento da ordem seguinte é feito com uma regulagem ou ferramental diferente do processamento da precedente. É composto pela perda da capacidade de produção, operação de troca ou ajuste de ferramenta, material para limpeza da máquina, perda de material, entre outros.

Muitas vezes o custo de preparação de máquina é considerado como parte do custo de processamento, logo não varia com a programação de produção e assim não é considerado na definição da mesma. Contudo, observa-se que em muitos sistemas de manufatura a

preparação da máquina varia muito conforme a relação precedente-sucessor entre as ordens de produção, o que torna claro que esse custo é afetado pela programação de produção e assim dever ser considerado na definição da mesma. Esses sistemas são chamados de dependentes da sequência.

• Custo de ociosidade da máquina. Ocorre quando uma ordem de produção programada para processamento, não está disponível para essa máquina. É composto pela perda de capacidade de produção, perda de capacidade de mão-de-obra, entre outros.

É claro que para problemas com uma máquina, onde as ordens de produção possam ser iniciadas no instante zero da programação, as ordens podem ser programadas uma em seguida da outra, evitando-se esse custo;

• Custo dos estoques. Ocorre quando o processamento total de uma ordem termina antes da sua data-de-entrega. É composto pelo custo financeiro de produto manufaturado, pelo custo de manter o produto armazenado (instalações, mão-de-obra, seguro, ...), entre outros;

• Custo por atrasos. É gerado quando o processamento total da ordem termina após a sua data-de-entrega. É composto pela possível perda do cliente, prejuizo à marca, multas contratuais, entre outros.

O custo total afeto pela programação de produção é uma combinação complexa desses custos. Dificilmente se terá uma boa representação do custo de um programa de produção com a consideração de um único componente desse, principalmente porque alguns desses custos são antagônicos e a programação da produção baseada em apenas um deles pode resultar em valores muito elevados de outros.

A decisão final de programação de produção é tomada pelo programador de produção e para tal esse profissional utiliza alguma ferramenta que lhe auxilie nessa decisão, além da sua experiência. Para tomar uma boa decisão de programação de produção é necessário que ele conte com uma ferramenta que:

• Reduza os custos envolvidos na programação de produção;

• Seja possível interferir na ponderação desses custos, para que com diferentes resultados ele possa adotar a programação que achar mais conveniente;

• Forneça informações que lhe permitam melhor negociar preços e prazos de entrega com os clientes.

1.1. Objetivo da Tese

O objetivo desta tese é a análise de Problemas de Programação da Produção em Uma Máquina, em um contexto com tempos de preparação dependentes da sequência e considerando uma medida de desempenho que pondere os custos de preparação de máquina, atraso e estoque, caracterizando um problema multiobjetivo. É apresentada uma heurística que utiliza técnicas de Busca Tabu para obter programas de produção para uma máquina, buscando a menor combinação desses custos. O sistema permite ao programador de produção alterar os pesos desses custos conforme suas necessidades.

1.2. Conteúdo do Trabalho

Fez-se aqui uma breve apresentação do tema do trabalho, assim como do objetivo almejado pelo mesmo. No capítulo dois faz-se uma apresentação dos principais conceitos e definições em problemas de programação da produção, assim como, em problemas de programação da produção em uma máquina, em problemas de programação da produção envolvendo datas de entrega e em problemas onde o tempo de preparação de máquina é dependente da sequência. São citadas, neste capítulo, várias referências que tratam de aspectos vários desses problemas. No capítulo três são apresentadas algumas metodologias aplicadas na resolução de problemas de programação da produção, dá-se um maior enfoque a metodologias que são utilizadas na heurística desenvolvida na tese. No capítulo quatro é apresentada a heurística desenvolvida durante o programa de mestrado do autor e que visa a resolução de problemas de programação da produção em uma máquina com tempos de preparação dependentes da sequência. Por último, o quinto capítulo apresenta os testes computacionais realizados com o emprego da heurística e as conclusões tiradas a partir dos mesmos.

Capítulo 2

Programação da Produção

Neste capítulo são apresentados alguns conceitos e definições em programação da produção, objetivando dar uma visão geral sobre o assunto e sobre o tema do trabalho. Não é intenção do presente capítulo aprofundar-se no assunto, mas sim dar uma visão necessária mínima para o entendimento da heurística. Para um estudo mais abrangente em programação da produção pode-se consultar Baker (1974), French (1982), Blazewicz et al (1992) e Pinedo (1995).

2.1. Visão Geral do Problema de Programação

Em problemas de programação da produção (PPP) estão envolvidas várias medidas de tempo. Algumas são consideradas na formulação desses problemas, outras são resultantes da resolução dos mesmos. Também devem ser consideradas na formulação desses problemas algumas hipóteses iniciais, as quais são úteis para a caracterização do problema e para gerar simplificações do processo real de produção, facilitando a resolução.

Sendo assim, são apresentadas inicialmente algumas definições e hipóteses, das mais empregadas em PPP, as quais podem ser assumidas para um problema genérico de programação de **n** ordens {1, 2, ..., n} a serem processadas em **m** máquinas {1, 2, ..., m}.

Antes de se apresentar as definições pertinentes a PPP, é necessário apresentar três conceitos que serão utilizados nessas definições:

- Máximo de um conjunto de quantidades: seja z_i qualquer quantidade pertencente a um

conjunto $Z = \{z_1, z_2, ..., z_n\}$, então:

$$Z_{max} = \max \{z_1, z_2, ..., z_n\};$$

• Média de um conjunto de quantidades:

$$\overline{Z} = 1/n \sum_{i=1}^{n} z_i$$

• $(expressão)^+$, significa que se o resultado de *expressão* > 0, $(expressão)^+$ assume tal resultado, caso contrário, assume o valor 0.

Definições Gerais para PPP:

• Operação (o_{ij}): processamento da ordem i na máquina j.

• Restrição tecnológica: o processamento de cada ordem nas máquinas deve respeitar uma sequência particular.

• Tempo de processamento (t_{ij}): duração de o_{ij}.

• Tempo de preparação de máquina (p_{ij}): tempo para ajuste, limpeza e troca de ferramental da máquina j para o processamento da ordem i.

• Instante de disponibilidade (r_i): instante no qual a ordem i torna-se apta para processamento.

• Data de entrega (d_i): instante em que a ordem i deve estar completada.

• Início de entrega (e_i): instante a partir do qual a ordem i pode ser completada, sem que se cause uma espera para entrega.

• Tempo de espera (a_{ik}): tempo que a ordem i fica sem processamento antes da k-ésima operação. a_{ik} = período de tempo entre a conclusão da operação (k-1) da ordem i em uma máquina j (ou r_i se k=1) e o início de processamento da operação k dessa mesma ordem em uma máquina j'.

• Tempo total de espera (A_i): soma de todos os tempos em que a ordem i ficou esperando por alguma operação.

$$A_i = \sum_{k=1}^{m} a_{ik} \qquad (2.1)$$

• Instante de conclusão (c_i): instante no qual a ordem i tem o processamento da última operação completado. O maior instante de conclusão, C_{max}, recebe uma denominação específica, *makespan*, e corresponde ao instante final da programação.

$$c_i = r_i + \sum_{k=1}^{m} (a_{ik} + t_{ij(k)}) \qquad (2.2)$$

• Fluxo de tempo (F_i): tempo que a ordem i leva para ser concluída.

$$F_i = c_i - r_i \qquad (2.3)$$

• Atraso (L_i): se a ordem i for completada após sua data de entrega, diz-se que ela está atrasada.

$$L_i = (c_i - d_i)^+ \qquad (2.4)$$

• Adiantamento (E_i): se a ordem i for completada antes do seu início de entrega ela está adiantada e tem que esperar até esse momento para ser entregue.

$$E_i = (e_i - c_i)^+ \qquad (2.5)$$

• Tempo de ociosidade (O_j): tempo que uma máquina j fica esperando por alguma ordem a ser processada nela.

$$O_j = C_{max} - \sum_{i=1}^{n} p_{ij} \qquad (2.6)$$

O gráfico de Gantt apresentado a seguir, figura 2.1, mostra como essas grandezas estão dispostas e interagem em uma programação de duas ordens em m máquinas. A sequência de processamento dada pelas restrições tecnológicas para a ordem 1 é (m-1, j, m, ..., 1, 2) e para a ordem 2 é (m, j, m-1, ..., 1, 2).

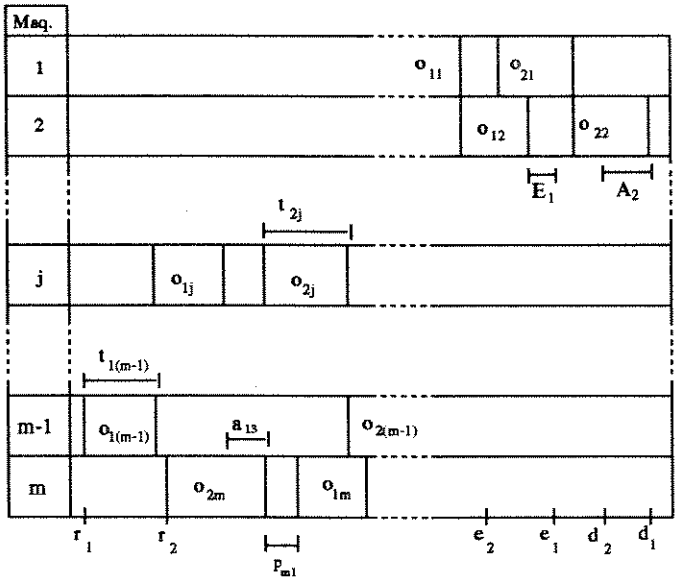

Fig. 2.1 - Gráfico de Gantt para 2 ordens em m máquinas

Hipóteses

Apresentam-se a seguir algumas hipóteses frequentemente consideradas em PPP:

- Inclue-se nos tempos de processamento, qualquer tempo necessário para preparação de máquina.
- As máquinas estão sempre viáveis, não se considera manutenção ou previsão de quebra. A mesma hipótese não é necessária para as tarefas.
- Nenhuma máquina pode processar mais de uma ordem ao mesmo tempo.
- Duas ou mais operações de uma mesma ordem não podem ser processadas ao mesmo tempo.
- Se uma operação já foi iniciada, esta deve ser completada antes que outra comece a ser processada na máquina (*no pre-emption*).
- Os tempos de processamento são independentes da sequência.
- Existe apenas um tipo de máquina para cada operação.
- Uma operação não pode ser quebrada (*split*) em mais de uma máquina.

Um problema de programação de n ordens em m máquinas é um problema extremamente complicado, já que para cada máquina existem n! permutações das ordens, assim para m máquinas tem-se $(n!)^m$ programas de produção possíveis. Devido às restrições tecnológicas o número de programas possíveis pode ser menor que $(n!)^m$.

A fim de se alcançar um dos objetivos, ou uma combinação de objetivos, adotam-

se medidas de desempenho para os mesmos. Em PPP medidas de desempenho são usadas para avaliação de programas de produção possíveis. A variação dos valores dessas medidas de desempenho expressa se se está aproximando ou distanciando dos objetivos almejados. Apresentam-se a seguir algumas das mais tradicionais medidas de desempenho empregadas em PPP.

Baseadas no Instante de Conclusão:

- Min F_{max} $\Rightarrow$ O custo do programa está diretamente relacionado com o tempo de conclusão das ordens.
- Min C_{max} $\Rightarrow$ O custo do programa depende da duração do processamento de todas as ordens (*makespan*).
- Min $\overline{F}$ $\Rightarrow$ O custo do programa está relacionado com a média de tempo que se leva para processar cada ordem.
- Min $\overline{C}$ $\Rightarrow$ O custo do programa está relacionado ao tempo médio para a conclusão de processamento das ordens.

Baseadas nas Datas de Entrega:

- Min $\overline{L}$ ou L_{max} $\Rightarrow$ Quando o custo do programa é definido pelas tarefas atrasadas.
- Max $\overline{E}$ ou E_{max} $\Rightarrow$ Quando se tem uma premiação por se terminar as ordens o mais cedo possível.
- Min N_t $\Rightarrow$ Quando o custo por se atrasar uma ordem não depende da magnitude do atraso, mas simplesmente se esta ordem está atrasada.

Baseadas no Custo de Inventário:

- Min $\overline{E}$ ou E_{max} $\Rightarrow$ Quando o custo do programa é determinado pelas ordens estocadas.
- Min $\overline{N_w}$ $\Rightarrow$ Minimizar o número médio de ordens esperando máquinas.

• Min $\overline{N_u}$ ⇒ Minimizar o número médio de ordens não completadas.

• Min $\overline{N_c}$ ⇒ Minimizar o número médio de ordens completadas; reduz os custos de inventário de produto final.

Baseadas na Eficiência das Máquinas:

• Max $\overline{N_p}$ ⇒ Maximizar o número médio de ordens sendo processadas em qualquer instante.

• Min $\overline{O}$ ou O_{max} ⇒ Minimizar o tempo médio ou máximo de máquinas ociosas.

Na prática, problemas de programação são bem mais complexos do que as possibilidades tratadas até aqui, onde as hipóteses feitas são bastante restritivas. Muitas dessas hipóteses não são confirmadas e outras podem ser necessárias. Para cada conjunto de hipóteses, caracteriza-se uma família de problemas de programação.

Alguns exemplos práticos para os quais tem-se a invalidação de algumas dessas hipóteses são:

• Tempos de preparação são independentes da sequência. Esta hipótese pode não ser verdadeira, pois os tempos de preparação de máquina, podem ser dependentes da sequência, pois existem muitas situações reais nas quais as perdas de tempo e material com a preparação da máquina entre uma ordem e outra, podem variar muito conforme a relação de precedência entre essas ordens.

• Se uma operação já foi iniciada, esta deve ser completada antes que outra seja iniciada na máquina. Na prática pode ser interessante e até mesmo necessário, se quebrar a execução de uma ordem em duas ou mais etapas.

Em casos reais, o custo total de um programa é uma combinação complexa de custos de processamento, de inventário, ociosidade de máquina, penalidades por atraso e outros. Logo, uma medida de desempenho que considere um único componente de custo, pode estar representando muito mal o custo total do programa. Raramente se consegue representar bem os

custos envolvidos na programação por funções tão simples como $\overline{C}$ ou L_{max}.

2.1.1. Programação em Uma Máquina

Exitem muitos trabalhos que se dedicam a problemas de programação da produção em uma máquina (PPPUM). Esses problemas têm uma complexidade menor que o problema geral de programação em m máquinas, já que o número máximo de programas de produção possíveis é n!, e não $(n!)^m$.

PPPUM pode ser visto como uma simplificação de problemas mais complexos. Esses problemas fornecem, em geral, um ótimo contexto para a análise e avaliação de medidas de desempenho e técnicas de resolução, as quais podem ser adaptadas e usadas posteriormente nesses problemas mais complexos.

Várias situações práticas também permitem a utilização dessa programação. Existem alguns sistemas de manufatura que envolvem uma máquina, como por exemplo na fabricação de espumas para colchões, ou na fabricação de mangueiras de borracha de diversas bitolas e cores. Em outros uma planta complexa deve ser tratada de forma agregada, funcionando assim como se fosse uma máquina. E em outros sistemas produtivos tem-se uma máquina agindo como gargalo do sistema, onde os tempos envolvidos nos processamentos desta máquina ditam o ritmo da produção.

Considerando-se um problema de programação em uma máquina, algumas das definições apresentadas devem ser alteradas:

- Tempo de processamento passa a ser t_i; tempo de processamento da ordem i;
- Operação passa a ser o_i , operação da ordem i;
- A máquina ficará ociosa apenas se não houver a possibilidade de se processar qualquer ordem (em função dos instantes de disponibilidade dessas);
- Não é mais preciso discriminar as máquinas no tempo de preparação, portanto p_i é o tempo de preparação da máquina para o processamento da ordem i.

2.2. Problemas com Prazos para Entrega

São vários os fatores que levam a determinação dos prazos para entrega, tais como: imposição do cliente, horizonte definido pela própria empresa, peças ou produtos intermediários que, em sistemas de "puxe" (*pull*), têm que estar prontos até suas datas de entrega para não comprometer processos subsequentes, etc. Seja qual for o fator que leve à determinação dos instantes para entrega, decisões são tomadas esperando-se que os mesmos sejam cumpridos. Caso isto não ocorra, muitos processos ou compromissos posteriores ficam prejudicados.

Esses instantes referem-se aos instantes de disponibilidade, início de entrega e data de entrega. Dependendo da estrutura do problema, considera-se, ou não, um, alguns ou todos esses. A estrutura do problema considera também a forma como são considerados.

O instante de disponibilidade pode ser igual a zero para todas as ordens, indicando que qualquer ordem pode ser processada a partir do início do horizonte de programação. Contudo, se esses instantes forem maiores que zero e diferentes entre si, deve-se incluir na programação um mecanismo que infactibilize a geração de programas de produção, onde o processamento de uma ordem comece antes do seu instante de disponibilidade.

O início de entrega e a data de entrega de uma dada ordem, determinam o intervalo de tempo em que o processamento dessa ordem deve ser completado. A violação desse intervalo de tempo, seja o processamento da ordem terminando antes do início de entrega, seja terminando após a data de entrega, pode ocasionar duas situações: infactibilidade do programa de produção ou penalidade.

Em muitas situações práticas, o início de entrega e a data de entrega referem-se ao mesmo instante e são considerados unicamente como data de entrega.

Sendo assim, redefine-se:

- Início de entrega = Data de entrega = d_i;
- Adiantamento $(E_i) = (d_i - c_i)^+$. (2.5)

A violação de datas de entrega acarreta vários custos, como:

- No contato com o cliente: chamada telefônica, papeis, tempo de pessoal gerencial, ...;
- Penalidades contratuais, caso existam;
- Perda da confiabilidade, implicando no risco de perda do cliente e prejuizo à marca;
- Expedição.

Existem muitos trabalhos que enfocam o problema de programação, onde os

atrasos são proibidos por restrições ou são considerados como penalidades nas medidas de desempenho. Sen e Gupta (1984) apresentam um *survey* (revisão) com muitas referências de tais trabalhos, as quais estão agrupadas em diferentes classes de medidas de desempenho e se aplicam a diferentes familias de problemas (uma máquina, máquinas em paralelo, ...)

Recentemente vêm se reconhecendo que os custos que afetam as decisões de programação e são gerados pelo desvio entre o instante em que a ordem foi completada e o instante de entrega, não são apenas custos por entrega com atraso, mas também custos por conclusão de processamento adiantada, antes do início de entrega, ou data de entrega, se esses referirem-se ao mesmo instante.

Segundo Fry et al (1987), Baker e Scudder (1990), Hoogeveen e van de Velde (1991), Lawrence (1991) e Cheng e Kahlbacher (1991), a consideração desses custos, gerados pelo estoque de produtos, foi intensificada com a adoção do conceito *just-in-time*, o qual prega que as ordens devem ser completadas o mais próximo possível das suas datas de entrega, a fim de se evitar custos de estocagem e custos de penalidades por atraso. É claro que *just-in-time* engloba um conjunto muito maior de princípios e não apenas aqueles relativos a data de entrega, contudo modelos de programação com penalidades por atrasos e adiantamentos parecem capturar a forma de programação em um enfoque *just-in-time*.

Pode-se definir então uma função de penalidade para um programa de produção P, f(P), que considera o início de entrega e a data de entrega referentes ao mesmo instante, como:

$$f(P) = \sum_{j=1}^{n} [\alpha_j (c_j - d_j)^+ + \beta_j (d_j - c_j)^+] \qquad (2.7)$$

Esta função pode ser adotada como uma medida de desempenho para a programação de n ordens, onde α_j é o peso pelo atraso da ordem j e β_j o peso pelo adiantamento da ordem j.

Os pesos das penalidades podem ser considerados de diferentes formas, resultando em diferentes funções de penalidade:

- $\alpha_j = \beta_j$, para $\forall$ j
- $\alpha_j = \beta_j$, para j = 1, ..., n
- $\alpha_j = \alpha$ e $\beta_j = \beta$, para $\forall$ j e $\alpha \neq \beta$
- $\alpha_j \neq \beta_j$, para $\forall$ j

Baker e Scudder (1990) apresentam um *survey* de trabalhos em programação da produção, cujas medidas de desempenho consideram diferentes possibilidades para os pesos por

adiantamentos e atrasos.

Quando se está considerando pesos diferentes para cada ordem de produção, está-se expressando o fato de que essas acarretam custos diferentes, seja por atraso ou por adiantamento, isto em função de aspectos específicos dessas ordens (exigem maiores cuidados na estocagem, ou têm multas contratuais por atraso diferenciadas, por exemplo) e em função de diferenças nos tamanhos delas (Hoogeveen e van de Velde 1991).

Os parâmetros α_j e β_j podem ser vistos como taxas de custos por atraso da ordem j e por estoque dos produtos referentes à ordem j, respectivamente, expressos em unidade monetária / tempo. Logo, a função de penalidade (2.7) tem como resultado, para um dado programa de produção, o custo total por atraso e estoque em uma certa unidade monetária.

Algums trabalhos consideram que todas as ordens têm uma mesma data de entrega, como em Szwarc (1989), Cheng e Kalbacher (1991), Sarin et al (1991), Panwalkar e Rajagopalan (1992) e Zheng et al (1993). Isto pode refletir a situação real onde muitos ítens constituem o pedido de um único cliente, ou reflete um ambiente de montagem no qual os componentes devam estar prontos ao mesmo tempo.

Existem formulações de problemas com penalidades por atraso e adiantamento, onde as datas de entrega não são dadas, mas sim consideradas como variáveis do problema. Cheng e Gupta (1989) relatam vários estudos de determinação de datas de entrega e Cheng (1991) estuda o problema de se determinar as datas de entrega e o programa de produção de n ordens, minimizando-se os desvios entre os instantes de conclusão das ordens e essas datas de entrega. Este enfoque reflete a prática de definir datas de entrega internamente, com o objetivo de guiar o progresso das atividades de chão de fábrica, ou para a negociação com o cliente, onde datas de entrega muito tarde são desincentivadas por um peso maior na função de penalidade, refletindo a situação real de ter que se oferecer descontos ou aumento do prazo de pagamento.

Assim como a data de entrega, outras quantidades podem ser inseridas na função de penalidade. No trabalho de Zheng et al (1993), além das penalidades por atraso e estoque este inclue uma penalidade por custo de fluxo de tempo, devido a inventário em processo. Panwalkar e Rajagopalan (1992) estudam o problema de programação cujo objetivo é se determinar os tempos de processamento (variáveis em função do esforço da máquina), uma data de entrega comum e o programa das ordens, afim de minimizar uma função de penalidade por atraso e adiantamento das ordens. Bai e Gershwin (1990) apresentam um algoritmo de tempo-real para a programação de ordens que visa minimizar os desvios dos instantes de conclusão das ordens

com as suas datas de entrega e manter o nível de inventário em processo o mais baixo possível.

Quando é grande o desejo de evitar desvios excessivos da data de entrega, costuma-se empregar o quadrado (ponderado ou não) desses desvios (Bagchi et al 1987).

Em PPPUM onde a medida de desempenho não seja regular, como é o caso de F(P), o emprego de tempos de ociosidade como variáveis é de grande importância, já que esses podem resultar em soluções com menores defasagens entre os instantes de conclusão e as datas de entrega de algumas ordens, diminuindo-se, no caso de F(P), custos com estoque de produtos acabados.

2.3. Dependência da Sequência

Em várias indústrias de processamento é comum o fato de uma máquina processar vários tipos de peças ou produtos. Nesses casos há a necessidade, na grande maioria das vezes, de se ajustar e limpar a máquina entre o processamento de uma ordem e outra. Esses ajuste e limpeza de máquina demandam tempo e geram custos, os quais são tratados por tempo e custo de preparação.

O custo de preparação é formado por custos envolvidos nas operações de limpeza e ajuste da máquina, tais como: material de limpeza, mão-de-obra, material perdido até a estabilização do processo de produção da nova ordem e perda da capacidade de produção.

Tanto o tempo como o custo de preparação de máquina entre uma ordem e outra ocorrem devido a diferenças nos seus processamentos, podendo esses ser maiores ou menores em função de uma maior ou menor relação entre as ordens e o processamento dessas. Logo, são dependentes da sequência das ordens gerada pela programação.

Portanto, esses custos e tempos de preparação de máquinas são afetados pela programação da produção, pela sequência das ordens definidas por essa. Sendo assim, é relevante a consideração desses custos e tempos na função medida de desempenho da programação, não apenas onde esses custos sejam da mesma grandeza que os custos por estoque e atraso, mas também em sistemas produtivos onde os custos de preparação sejam bem inferiores aos custos de estoque e atraso, pois mesmo nestes casos os tempos de preparação de máquina influenciam muito atrasos e adiantamentos geradores desses custos. Manson e Anderson (1991) pregam que a não consideração desses em heurísticas de programação, é motivo para o fraco desempenho

delas em situações práticas.

Definindo-se como p_{ij} o tempo de preparação de máquina quando a ordem i precede imediatamente a ordem j, pode-se redefinir a função de penalidade (2.7) para que esta também considere o custo de preparação de máquina,

$$f(P) = \sum_{k=1}^{n+1} [\gamma_{j_{k-1}j_k} p_{j_{k-1}j_k} + \alpha_{j_k}(c_{j_k} - d_{j_k})^+ + \beta_{j_k}(d_{j_k} - c_{j_k})^+] \quad (2.8)$$

onde, $\gamma_{j_{k-1}j_k}$ pode ser visto como 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 / tempo. Expressões do tipo $(x)^+$ indicam que se $x \geq 0$ adota-se o valor de x caso contrário adota-se x=0.

Observando-se o intervalo de variação dos tempos de preparação na expressão (2.8) nota-se a consideração de um estado inicial e final de ajuste da máquina, denotado como ordem **0** (n+1 = 0), de onde o programa de produção deve partir e para onde deve voltar. Sendo assim, essa ordem zero irá ocupar sempre a mesma posição na sequência e implica nos tempos de preparação p_{0j} e p_{j0}. Logo, quando é feita tal consideração passa a se trabalhar com um problema de n+1 ordens.

O problema de programação com tempos de preparação dependentes da sequência e ordem zero, pode ser visto no contexto de problemas de roteamento de veículos, especificamente onde tem-se o Problema do Caixeiro Viajante (PCV) (Bianco et al 1988), (Gendreau et al 1991), (Mason e Anderson 1991) e (Laporte 1992). O processamento de uma ordem j corresponde à visita a um cliente j, a qual leva um tempo t_j e onde p_{ij} é o tempo que o caixeiro leva para percorrer o percurso entre o cliente i e o cliente j. O estado inicial e final de ajuste de máquina no PPPUM, ordem zero, corresponde à cidade de origem e chegada do caixeiro.

Se forem estabelecidos instantes para o início e conclusão do processamento no PPP em uma máquina com tempos de preparação dependentes da sequência e com restrições de períodos para entrega, este problema pode ser considerado equivalente ao problema do Caixeiro Viajante com Janelas de Tempo (PCVJT), onde esses instantes estão associados aos instantes que definem os intervalos de tempo para visita aos clientes no problema do Caixeiro Viajante com Janela de Tempo (Solomon 1987).

As restrições referentes ao instante a partir do qual a conclusão do processamento

da ordem não incorre em tempos de adiantamento e ao instante até o qual a conclusão do processamento da ordem não incorre em tempos de atraso, estão associadas às janelas de tempo no PCVJT. Portanto, vários problemas desse tipo podem diferir: 1) quanto à medida de desempenho (tempo máximo para o processamento das ordens - visita aos clientes; atraso médio; ponderação dos custos; e outros), 2) quanto à estrutura da matriz de tempos de preparação - viagem (simétrica; assimétrica) e 3) na largura da janela de tempo (com ou sem limitante[2] superior, ou inferior; ou se ambos são iguais - apenas para o caso de penalidade).

Nem toda troca de ordem implica em ajuste e limpeza na máquina. Podem existir ordens diferentes (diferentes clientes, ou datas de entrega) que possuam as mesmas características de processamento, ou suas diferenças não implicam em operações de preparação entre essas. Sendo assim, se essas ordens são programadas uma em seguida da outra, não se tem nem tempo nem custo de preparação de máquina e essas ordens podem ser agrupadas em uma mesma família.

O objetivo das empresas é atender ao mercado consumidor da melhor forma, ou seja, atender a necessidade por produtos. Essa necessidade é cada vez mais diversificada, pois existe uma gama enorme de preferências e de capacidade de consumo. Portanto, as empresas tentam cada vez mais aumentar o leque de opções de produtos para o consumidor.

Contudo, esse aumento na variedade de produtos aumenta muito a complexidade da tarefa de programação, devido a maior quantidade de ordens a serem programadas. Aumenta também os custos com preparação de máquina devido a diversidade de produtos. A fim de se diminuir tais custos deve-se projetar produtos e peças agrupados em famílias com afinidades de processamento.

Segundo Manson e Anderson (1991) e Woodruff e Spearman (1992), o agrupamento de ordens em famílias aumenta a capacidade de se reduzir tempos e custos de preparação de máquina, pois esses tempos e custos ou não existem ou são muito reduzidos entre ordens de uma mesma família. A programação das ordens agrupadas em famílias resulta no estudo de tecnologia de grupo, onde esses agrupamentos devem ser considerados e obtidos do projeto do produto e do planejamento do processo. Zdrzalka (1991) também considera o agrupamento em família para o seu algoritmo, contudo ele considera um único tempo de preparação entre famílias de ordens, o que raramente ocorre na prática, onde tem-se tempos e

---

[2] Limitante refere-se a instantes que não podem ser violados; caso isso ocorra, a solução é infactível.

custos de preparação diferentes para diferentes relações de precedência e sucessão entre famílias.

Woodruff e Spearman (1992) apresentam uma outra forma de agrupamento, essa não referente aos tempos de processamento, mas a importância do cliente, ou do pedido desse, para a empresa. São criadas então duas classificação de ordens: as ordens obrigatórias, referentes a pedidos importantes e que devam ser cumpridos e ordens opcionais, referentes a pedidos não tão importantes ou a previsões de venda ainda não confirmadas, essas podem ou não estar no programa de produção. Os autores consideram a infactibilidade para atrasos e não penalidades.

Essa diferença de ordens pode ser considerada na função de custos, onde os custos de atraso devem ser diferentes para ordens obrigatórias e ordens opcionais. As ordens dos clientes preferenciais devem ter um custo de atraso maior que as dos clientes não preferenciais.

Um outro trabalho que trata PPPUM com tempos de preparação dependentes da sequência e também prioriza ordens referentes a alguns clientes é apresentado por Lima (1993). Um ponto muito interessante desse trabalho, está na aplicação de Lógica Nebulosa para a resolução do problema.

Capítulo 3

Métodos de Resolução

A programação da produção é um dos problemas mais estudados por profissionais que se dedicam a atividade de manufatura. Esses estudos vêm resultando na determinação de inúmeras variações desse problema, variações essas que se dão em função de características distintas de cada sistema de manufatura. Resultam também na aplicação e até mesmo desenvolvimento, de diversas metodologias para a resolução dessas variações.

Neste capítulo são apresentadas algumas dessas metodologias. Devido a complexidade do problema enfocado está se apresentando métodos não exatos, que embora não garantam que a solução encontrada seja a melhor, conseguem boas soluções sem grandes demandas de tempo e memória computacionais. Será dada uma maior atenção a apresentação da metaheurística de Busca Tabu, usada na proposta de resolução deste trabalho.

Devido a correlação do problema de programação em uma máquina com tempos de preparação dependentes da sequência com o PCV, muitos dos métodos empregados para a resolução de um são empregados para a resolução do seu correlato. Inclusive, a heurística proposta nesta tese teve como *insight* a metodologia apresentada por Pureza (1990), para a resolução de Problemas de Roteamento de Veículos.

Um estudo mais aprofundado a respeito dos métodos que serão apresentados a seguir, assim como de alguns outros aqui não enfocados, podem ser vistos em Bodin et al (1983), Solomon (1987), Desroches et al (1988) e Laporte (1992). Caso o objetivo seja o de se conhecer mais trabalhos que tratem do problema de roteamento de veículos com janela de tempo, Solomon e Desrosiers (1988) apresentam um *survey* de trabalhos que abordam esse problema.

3.1. Métodos Exatos

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

Para tais problemas os algoritmos exatos requerem um número de passos que cresce como uma função exponencial do número de nós a serem visitados (clientes no caso do PCV, ordens no caso do PPP). E ao se acrescentar restrições ao mesmo, tal como de janela de tempo, a resolucão de tais problemas por métodos exatos torna-se ainda mais difícil (Solomon 1987).

Muitos métodos exatos foram e continuam sendo elaborados para a resolução do PCV e do PPP, esses podem ser encontrados nas referência já citadas. Contudo, a complexidade desses problemas, faz com que a maior parte dos algoritmos que possam ser reproduzidos, sejam aplicáveis apenas para problemas de até 100 nós.

3.2. Métodos Aproximados

Para resolver problemas grandes com eficiência tem-se que empregar técnicas de resolução que não garantam a melhor solução, mas que garantam quase sempre uma boa solução, não empregando para tal grandes quantidades de tempo e memória computacionais. É com este intuito que vem se desenvolvendo o uso de heurísticas, que melhoram bastante a eficiência da programação, sacrificando idéias de perfeição.

3.2.1. Métodos de Construção

Essas heurísticas são geralmente usadas para gerar uma rota no PCV, ou rotas no Problema de Roteamento de Veículos. Elas são empregadas em problemas onde o objetivo é minimizar a distância total de viagem. Em PPP geram programas de produção utilizando a matriz de tempos de preparação, onde o objetivo é o de se minimizar o tempo total da programação. Se o objetivo do problema for diferente deste, outros fatores devem ser considerados na construção do programa.

a) Algoritmo do Vizinho Mais Próximo

É um algoritmo míope, pois uma rota possível é construído tomando-se em cada passo a decisão que é imediatamente a mais vantajosa. No contexto PCV

Passo 1 - Tome um nó arbitrariamente como ponto inicial.

Passo 2 - Dentre os nós não roteados, determina-se o nó mais próximo ao último nó e inclua esse na rota.

Passo 3 - Una o último nó da rota ao primeiro. Volte a 2.

b) Algoritmo de Inserção

No contexto de programação, um algoritmo de inserção determina qual o melhor nó ainda não programado a ser inserido em um sub-programa[3] e então determina onde esse deve ser inserido.

Passo 1 - Inicie um sub-programa constituido de apenas dois nós. Se o nó inicial for considerado nó zero (estado inicial e final de preparação de máquina), o sub-programa deve ser constituido desse nó e um outro escolhido por algum critério.

Passo 2 - Passo de Seleção. Encontre um nó k ainda não programado com base em um dado critério, por exemplo:

- que produza o menor aumento no tempo total do sub-programa;
- que possua o menor tempo de preparação com qualquer nó do sub-programa;
- que possua o maior tempo de preparação com qualquer nó do sub-programa;
- arbitrariamente.

Passo 3 - Passo de Inserção. Insira k entre i e j tal que $p_{ik} + p_{kj} - p_{ij}$ seja o menor para os nós i, j adjacentes já pertencentes ao sub-programa.

Passo 4 - Volte para o passo 2 a menos que os nós já tenham sido todos inseridos.

3.2.2. Métodos de Melhoria

Em um PPPUM com tempos de preparação dependentes da sequência, uma solução factível é qualquer programa que satisfaça as restrições impostas. Pode-se aplicar a uma solução

---

[3]Denomina-se sub-programa, um programa parcial com k nós de um programa total de n nós (k < n).

desse problema um procedimento heurístico que produza outras soluções **"vizinhas"** a essa, onde a maioria das suas posições são ocupadas pelas mesmas ordens.

Métodos que aplicam a uma dada solução **mecanismos de perturbação**, gerando outras soluções vizinhas a essa, e buscando soluções com melhores valores de uma medida de desempenho adotada, são denominadas genericamente de métodos de melhoria, ou de busca local.

A obtenção da solução de partida se dá por qualquer heurística de construção de programas, ou mesmo através de uma sequência aleatória das ordens. Emprega-se, então à essa solução inicial mecanismos de perturbação, gerando outros programas vizinhos a ela. Encontra-se, então, o programa com melhor valor na vizinhança gerada. Se esse for melhor que a solução anterior, ele passa a ser a nova solução.

Os mecanismos de perturbação são aplicados sucessivamente até que se atinja um ótimo local. Seja X o espaço de soluções de um PPP, $x \in X$ uma solução desse problema e c(x) o custo que a solução x acarreta (pode ser obtido pela expressão 2.8, por exemplo). A aplicação de um mecanismo de perturbação MP em x gera uma vizinhança V(x) dessa solução, onde essa vizinhança é um sub-conjunto do espaço de soluções, $V(x) \subset X$. Define-se $x^0 \in V(x)$ como a solução com menor custo na vizinhança de x e $x^* \in X$ como a solução com menor custo já obtida.

Os métodos de melhorias diferem pelos procedimentos de geração da vizinhança. Para diferentes métodos de perturbação aplicados a uma solução geram-se diferentes vizinhanças dessa solução, como representado na figura 3.1 a seguir.

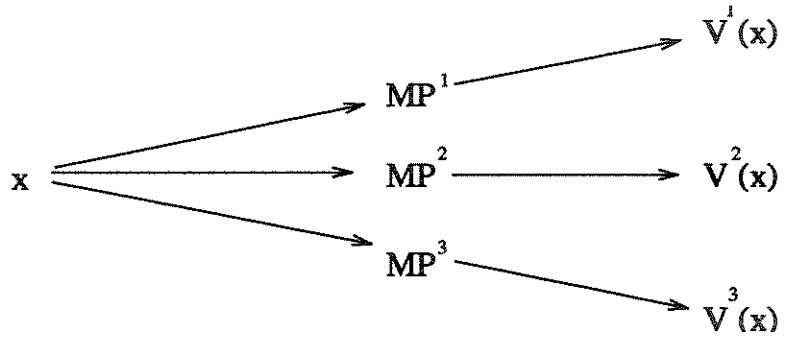

Fig. 3.1 - Vizinhanças e Métodos de Perturbação

O procedimento dos métodos de melhoria pode ser representado pelo fluxograma da figura 3.2 a seguir.

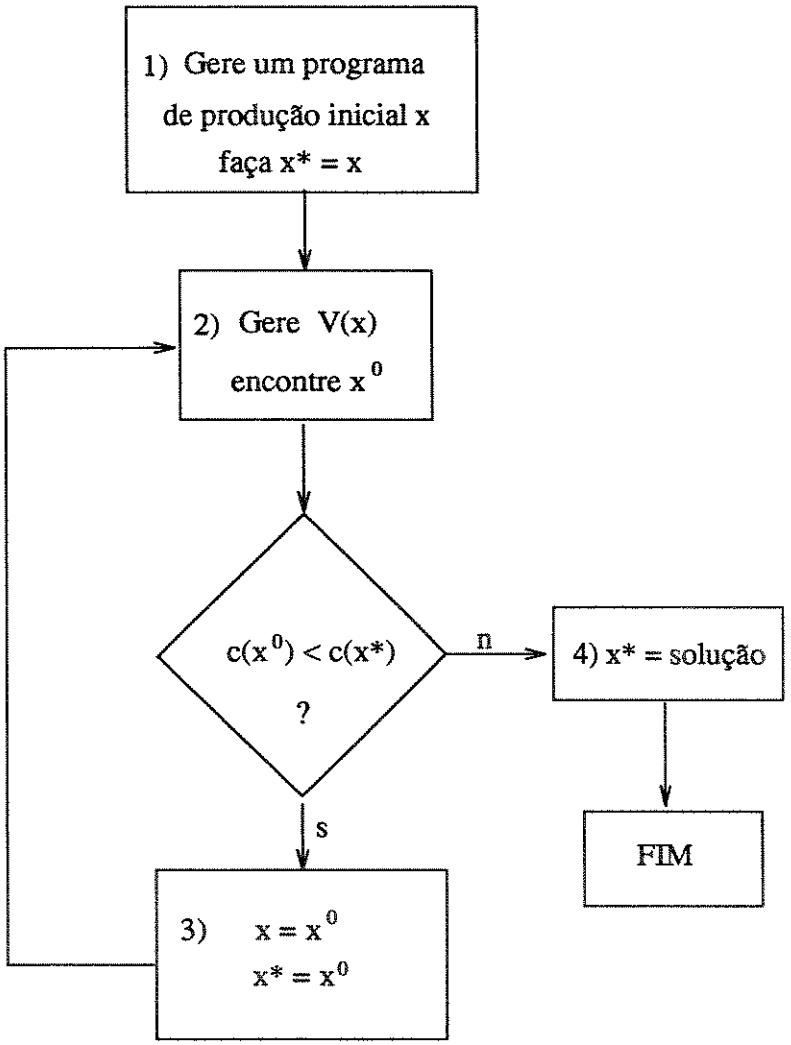

Fig. 3.2 - Métodos de Melhoria

No contexto do PCV, dois métodos clássicos de melhoria são troca simples e r-opt.

a) Troca Simples

Método de melhoria onde a vizinhança de uma solução é gerada trocando-se pares de clientes de lugar.

Dado um programa A-B-C-D-E-A, um dos vizinhos gerados por esse método é o programa A-D-C-B-E-A, obtido pela troca de posição entre o nó B e o nó D, como pode ser visto na figura 3.3. A vizinhança dessa solução é formada por todo programa que possa ser obtido com a troca de lugar de dois nós quaisquer dela.

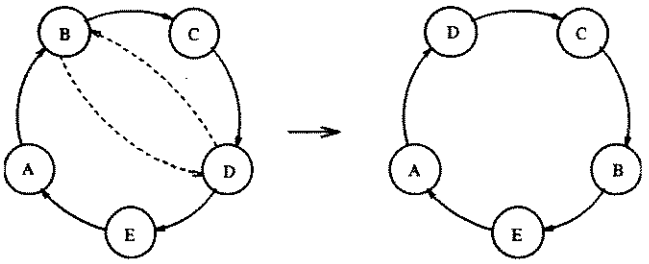

Fig 3.3 - Troca Simples

b) r-opt

Aqui a vizinhança de uma solução é gerada removendo-se **r** arestas não adjacentes e reconectando-se as r cadeias remanescentes em todas as formas possíveis.

Diz-se que o programa resultante do método de melhoria r-opt é r-ótimo. Para n nós, considerar todas as possíveis trocas envolve um esforço computacional da ordem de n^r, o que tem limitado o valor de r a 2 ou 3 (Pureza 1990) . Para o caso 2-opt existe apenas um modo de se reconectar as 2 cadeias resultantes. A figura 3.4 mostra como um vizinho de uma solução é gerado por esse método.

Removendo-se as arestas A-B e C-D produz-se as cadeias B-C e D-E-A, as quais podem ser reconectadas pela "criação" das arestas A-C e B-D gerando-se um novo programa A-C-B-D-E-A.

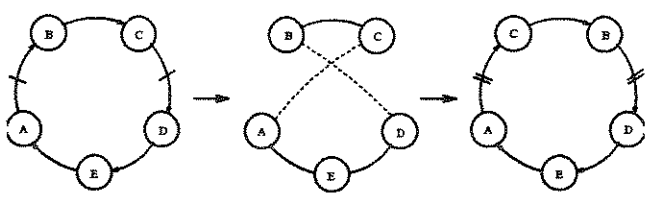

Fig. 3.4 - Geração de Vizinhança com 2-opt

Observa-se portanto que os métodos de melhorias resultam em um ótimo local (1º encontrado), não necessitando para tal de grande esforço computacional. A qualidade dessa solução depende da solução inicial encontrada, assim como da topologia do espaço de soluções gerado para o problema com o mecanismo de perturbação adotado.

Caso o espaço de soluções do problema tenha poucos mínimos locais, ou caso eles não diferirem muito em profundidade, os métodos de busca local podem dar bons resultados.

Contudo, isso pode não ocorrer, sendo então, interessante sair desses ótimos locais para analisar outras regiões. Para não ficar preso em um ótimo local e ampliar o espaço de busca, atingindo-se soluções quase ótimas, desenvolveram-se metódos capazes de transpor a otimalidade local mais robustos e eficientes. Alguns desses são: *Simulated Annealing*, Busca Tabu, Algorítmos Genéticos.

a) Simulated Annealing

Método de melhoramentos sucessivos derivado da analogia com o processo de *annealing* (têmpera) de materiais usado na mecânica (Eglese 1990) e (Ogbu e Smith 1990).

Proposto por Metropolis et al (1953) para determinar as propriedades de ligas metálicas, Simulated Annealing vem despertando maior interesse desde a apresentação dos trabalhos de Kirkpatrick et al (1983) e Cerny (1985), onde eles mostram como um modelo de Simulated Annealing de sólidos pode ser usado para problemas de otimização.

A fim de se trazer um material a um estado sólido de mínima energia, é necessário aquece-lo até suas partículas estarem distribuidas aleatóriamente no estado líquido. Então para se evitar cristalização em estados intermediários (ótimos locais) de energia, essa temperatura é gradualmente reduzida em passos, até o sistema atingir um equilíbrio para um dado nível de temperatura.

Em uma temperatura alta T, todos os possíveis estados podem ser atingidos, mas como o sistema "esfria", o número de possibilidades diminue e o processo vai sendo reduzido até um estado estável.

Em otimização combinatorial o objetivo é se mover de uma solução inicial a uma solução de mínimo custo. Seja T o estado do processo (T corresponde a um nível de temperatura em um sistema físico). No início o valor de T é alto e o número de movimentos permitidos também é alto. Esse número decresce com T, até nenhuma mudança ser possível. Atingiu-se então um mínimo local.

No início o algoritmo é similar a um método de melhoria, porém a substituição de uma solução x por um vizinho x' é permitida algumas vezes mesmo se essa substituição provocar um aumento na função de custo, chamado de movimento de piora. A probabilidade de se aceitar uma substituição que cause um movimento de piora é expressa pela função $\exp(-\delta / T)$, onde δ é o aumento da função de custo e T é um parâmetro de controle (temperatura) que decresce no decorrer da resolução, implicando assim na diminuição da aceitação de uma substituição que aumente o custo.

3.3. Busca Tabu

Busca Tabu (Glover 1989), (Glover 1990) e (Glover et al 1993) é um método que permite transpor limitações de otimalidade local, permitindo movimentos que piorem o valor da medida de desempenho. Afim de previnir ciclagens e estimular a exploração de novas regiões, deve-se proibir, por um certo número de iterações, a realização de movimentos que possam gerar soluções recentemente exploradas.

Nascida no final da década de 70 (Glover 1977), Busca Tabu tem sido aplicada a problemas de diversos contextos, como Roteamento de Veículos (Pureza 1990), Dimensionamento de Lotes (Berretta - 1993) e Programação da Produção (Widmer e Hertz 1987) e (Woodruff e Spearman 1992) e ainda no Projeto de Circuitos, Telecomunicações, Grafos, Redes Neurais e outros como levantado por Glover et al (1993).

Como já mencionado, métodos de melhoria partem de uma solução inicial e aplicando mecanismos de perturbação realizam movimentos que promovem a geração de soluções melhores. Esses mecanismos de perturbação são aplicados à solução corrente até que se atinja um ótimo local. Como, em geral, não se pode garantir que esse ótimo é global, ou mesmo uma boa solução, é desejável continuar o processo de busca, explorando mais o espaço de soluções.

Para continuar o processo de busca, a partir de um ótimo local, há a necessidade de se realizar um movimento de piora. Ao se aplicar o mecanismo de perturbação nessa nova solução, pior que a anterior, muito provavelmente está se realizando o movimento reverso ao de piora e a solução vizinha será novamente o mínimo local já explorado. Caracteriza-se então uma

ciclagem composta por duas soluções, a gerada por um movimento e a gerada pela reversão deste.

Para se evitar a geração de soluções já visitadas através de movimentos de reversão, forçando-se assim a busca em novas regiões , pode-se proibir certos movimentos de serem feitos.

Contudo, conforme o processo de busca avança, o número de movimentos proíbidos torna-se muito grande, e as soluções encontradas vão ter uma configuração muito diferente daquela referente ao primeiro ótimo local encontrado. Esse carater excessivamente restritivo pode prejudicar a investigação de muitas regiões que podem conter boas soluções e que poderiam ser acessadas pelo emprego desses movimentos tabus. A possibilidade de retorno a uma dada solução após sucessivos movimentos é muito pequena. Sendo assim, é interessante que a proibição a certos movimentos desapareça, após algumas iterações.

Esses três aspectos abordados :

- Possibilidade de movimentos de piora,
- Proibição de movimentos,
- Liberação de movimentos proibidos após um certo número de iterações,

constituem os elementos básicos da Busca Tabu.

Um processo simples de busca pode ser representado como na figura 3.5 a seguir, onde **it** indica a iteração atual do processo de busca, a qual tem valor inicial zero e um valor máximo chamado **limite**, Ao se alcançar esse valor limite o processo de busca é encerrado.

Os movimentos proibidos, movimentos tabus, podem ser representados por atributos. Esses atributos são armazenados em uma estrutura denominada **lista tabu**. Essa estrutura, que pode ser outra que não uma lista, é consultada quando da realização de movimentos para a geração da vizinhança, a fim de se determinar quais soluções pertencentes a essa vizinhança não devem ser consideradas, evitando ciclagens e forçando a busca em novas regiões.

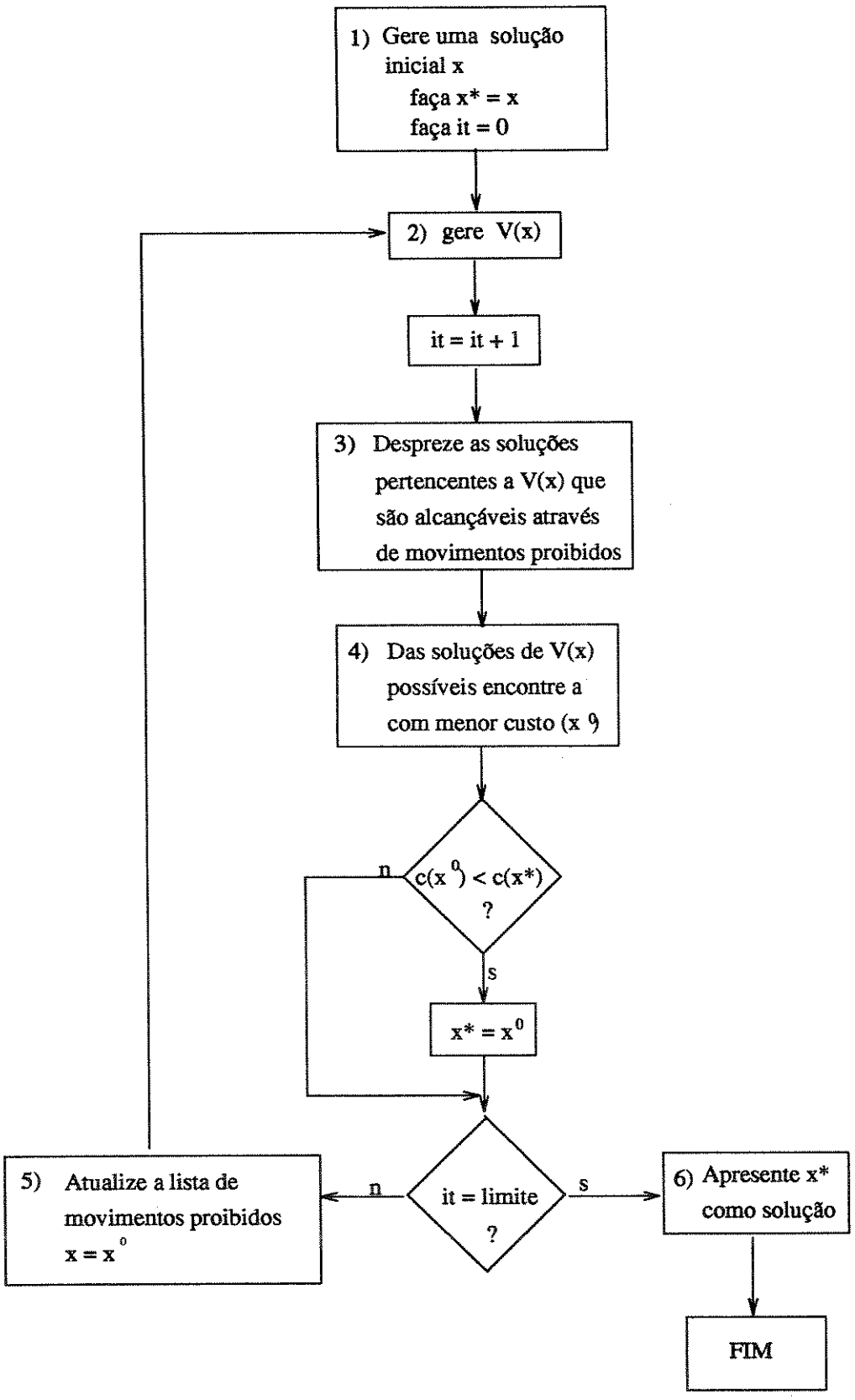

Fig. 3.5 - Busca Tabu Simples

3.3.1. Lista Tabu

Como já analisado, para se evitar ciclagens e possibilitar a busca por novas regiões é necessário proibir a reversão de movimentos feitos recentemente e para se possibilitar que mais regiões do espaço de busca sejam acessadas, dando maior flexibilidade ao processo de busca, é necessário também que esses movimentos sejam proibidos apenas por um número adequado de iterações. Desta forma, defini-se lista tabu como uma estrutura que armazena atributos que caracterizam os movimentos.

Dependendo do atributo escolhido pode-se proibir não apenas um único movimento, mas um conjunto de movimentos associados a esse atributo, podendo a busca tornar-se mais ou menos restritiva, dependendo da seleção dos atributos e do número deles.

Como exemplo, considere o método de melhoria 2-opt. Os movimentos que geram um novo vizinho são os de se eliminar duas arestas e se reconectar as 2 cadeias restantes usando-se duas outras arestas. Esse movimento pode ser representado, portanto, pelas arestas eliminadas e pelas adicionadas. Adotando-se como atributo as arestas eliminadas, deve-se proibir, por um certo número de iterações, movimentos em que essas arestas apareçam sendo adicionadas. Deve-se definir um número mínimo de arestas presentes para que o movimento seja declarado tabu. Se esse número for 1, basta que uma aresta a ser adicionada em um movimento coincida com uma aresta já eliminada em um movimento anterior, para implicar em movimento tabu. Isto é mais restritivo se duas arestas adicionadas no movimento tiverem que coincidir com duas arestas eliminadas em movimentos anteriores que estejam na lista tabu.

A cada iteração a lista é consultada, se um movimento é declarado tabu ele não é realizado. A cada iteração realiza-se também o processo de atualização da lista, que consiste da inserção de novos atributos a serem proibidos e da liberação dos atributos que nela permaneceram pelo número de iterações indicado.

Portanto, o tamanho da lista, que define o número de iterações em que os atributos serão considerados tabu, é de importância fundamental para se conseguir evitar ciclagens, pois se a lista for muito pequena a reversão de algum movimento pode fazer com que o processo se direcione para soluções já exploradas. O tamanho da lista também não pode ser muito grande, pois assim se reduziria as possibilidades de escolha, e com essas o acesso à soluções promissoras. Segundo Berretta (1993) o tamanho da lista é um parâmetro que depende do algoritmo no qual será incorporada a lista tabu e do problema que está sendo resolvido.

Se a estrutura da lista tabu for realmente uma lista, o atributo recém-declarado como tabu será inserido na última posição dessa lista e o atributo que estiver na primeira posição dela será liberado, isso se a iteração corrente for maior que o tamanho da lista. Caracteriza-se assim uma estratégia FIFO. O problema com este tipo de estratégia é que a cada movimento candidato tem-se que consultar todos os elementos da lista para se saber se há atributo(s) dele na lista, o que torna tal consulta lenta.

Para evitar tal problema pode-se armazenar os atributos tabus em uma matriz, onde cada posição irá representar um atributo, associando ao mesmo um valor que represente a iteração da sua liberação. Se a posição referente a um determinado atributo possuir um valor inferior a iteração atual, este atributo não é tabu. Logo, o movimento que este representa pode ser executado. Para se proibir um atributo, deve-se inserir na posição da matriz referente ao mesmo, um número que indique até qual iteração este deve ser considerado tabu. Este número é obtido pela adição da iteração atual com o número de iterações que o atributo ficará proibido, chamado de *Tabu Tag*.

Além da facilidade no processo de consulta, o armazenamento em matriz permite a utilização de diferentes números de iterações proibidas para os atributos, gerando um tamanho de lista aleatório. O uso de tamanhos de lista aleatórios tem se mostrado mais vantajoso para evitar ciclagens, já que muitas dessas possuem periodicidade, e tendem a se repetir de período em período (Taillard 1990).

Os tabu tags, em matrizes que utilizam tamanho de lista aleatório, são definidos em um intervalo de variação e será a magnitude e a amplitude desse intervalo que irá ditar a eficiência do algoritmo em evitar ciclagens e promover a busca em novas regiões, da mesma forma que o tamanho de lista em estruturas FIFO.

3.3.2. Aumentando a Capacidade do Processo de Busca

A escolha da estrutura de armazenamento dos atributos e o mecanismo de geração de movimentos adotado, são os mecanismos fundamentais em um processo de busca tabu.

Um mecanismo de busca tabu simples é composto basicamente por esses dois mecanismos. Contudo, conforme foram sendo feitos estudos sobre a proposta da busca tabu e foram sendo feitas implementações dessa para a resolução de vários problemas, observou-se que

outros mecanismos poderiam ser acoplados a esta para que o processo de busca produzisse uma maior eficiência. Alguns desses mecanismos são: Níveis de Aspiração, Oscilação Estratégica e Funções de Médio e Longo Prazo.

Níveis de Aspiração

Dificilmente ao se escolher um atributo que represente e proiba um movimento, este não proiba alguns outros movimentos que compartilhem desse mesmo atributo. Eliminam-se assim algumas opções de movimentos que poderiam fornecer soluções promissoras até então não consideradas.

O Critério de Aspiração foi elaborado para se desconsiderar a condição tabu de movimentos que promovam a busca em uma região atrativa, aumentando a flexibilidade do processo de busca.

A implementação do critério de aspiração emprega uma função que avalia um movimento proibido, informando se este é ou não um movimento atrativo. É claro que o emprego do critério de aspiração deve ser feito evitando-se a ocorrência de ciclagens.

Oscilação Estratégica

A busca tabu tenta prevenir ciclagens e promover a exploração de diferentes espaços de solução. Nesse processo ela apresenta o aspecto de induzir o comportamento da busca, fazendo com que sejam criadas sucessivas soluções, nas quais o valor da função objetivo varia, crescendo e decrescendo. Essa oscilação é resultado da combinação da orientação agressiva da busca que escolhe sempre o melhor vizinho e do elemento restritivo representado pela lista tabu.

Esta indução à busca pode ser aperfeiçoada empregando-se mecanismos que diversifiquem a natureza das regiões exploradas aumentando a flexibilidade e melhorando o direcionamento do processo de busca, tais mecanismos são tratados como Oscilação Estratégica. Um exemplo desses é a exploração de regiões infactíveis e o posterior retorno a uma região factível, na tentativa de explorar regiões factíveis melhores.

Funções de Memória de Médio e Longo Prazo

Um processo de busca tabu simples emprega uma função de memória de curto prazo, representada pela lista tabu. Essa função de memória trabalha com a recência de atributos de movimentos. Ela não considera a qualidade desses atributos e não permite grandes alterações do espaço de soluções. Esses dois aspectos são considerados em funções de memória de médio e longo prazo.

Funções de memória de médio prazo objetivam a intensificação da busca em espaços de soluções caracterizados por atributos presentes, ou que se repetiram, na maioria das melhores soluções encontradas até o momento.

Funções de memória de longo prazo promovem a diversificação dos espaços de solução da busca. Através de critérios de avaliação que podem ser usados por um processo de busca heurística, essas funções guiam o processo para espaços de solução marcadamente contrastante com os examinados até o momento.

Capítulo 4

Heurística para Resolução do Problema

Este capítulo apresenta uma heurística para a resolução do problema de programação da produção em uma máquina com tempos de preparação dependentes da sequência. Inicialmente são feitas algumas considerações a respeito de hipóteses que delimitarão o problema.

A heuristica é dividida basicamente em três etapas:

- Obtenção da solução inicial;
- Melhoria da solução;
- Busca Tabu.

A obtenção da solução inicial, primeiro programa de produção a ser definido, é conseguida usando-se uma adaptação da heurística de inserção apresentada por Solomon (1987). Nessa heurística, Solomon utiliza um critério que pondera , na inserção de cada nó ainda não programado, o aumento da distância e do tempo que cada inserção provoca e o tempo de espera para o atendimento do nó a ser inserido.

A etapa de melhoria da solução inicial até o primeiro ótimo local é realizada utilizando-se o método de melhoria 2-opt. Com essa heurística faz-se todas as trocas de pares de arestas não adjacentes em uma solução, gerando-se assim soluções vizinhas à solução corrente. Se o custo do melhor programa vizinho for menor que o custo da solução corrente, a solução do

problema passa a ser o melhor vizinho. Esse processo se repete até que o custo da solução corrente seja menor que o de qualquer vizinho seu

A etapa de busca por melhores soluções será realizada pela meta-heurística de busca tabu. Nessa etapa é usada uma busca tabu simples, sem a utilização de níveis de aspiração, oscilação estratégica ou funções de memória de médio ou de longo prazo. A estrutura do limitante de movimentos é uma matriz, onde cada elemento indica até qual iteração o atributo em questão está proibido.

A etapa de Busca Tabu poderia ser aplicada já à solução inicial, contudo, utilizando um método de busca local antes desta, tem-se a oportunidade de confrontar os resultados obtidos aplicando-se os dois métodos, observando-se assim o aumento da eficiência da heurística com o emprego da busca tabu sobre a possibilidade dessa heurística não utilizá-la.

O mecanismo de perturbação usado na busca tabu para geração de soluções vizinhas a serem analisadas também é o 2-opt. Com o 2-opt gera-se a melhor solução vizinha a da iteração atual, desprezando-se as soluções obtidas com movimentos tabu, até que um critério de parada seja atingido.

A figura 4.1 mostra, através de um fluxograma simples, como é o funcionamento da heurística e como se dá a integração entre os procedimentos que a compõem.

Essa figura faz uma representação a nível estratégico da heurística. Isto quer dizer que os procedimentos nela citados são compostos de outros procedimentos. Um maior detalhamento desses procedimentos é dado quando da apresentação de cada etapa especificamente.

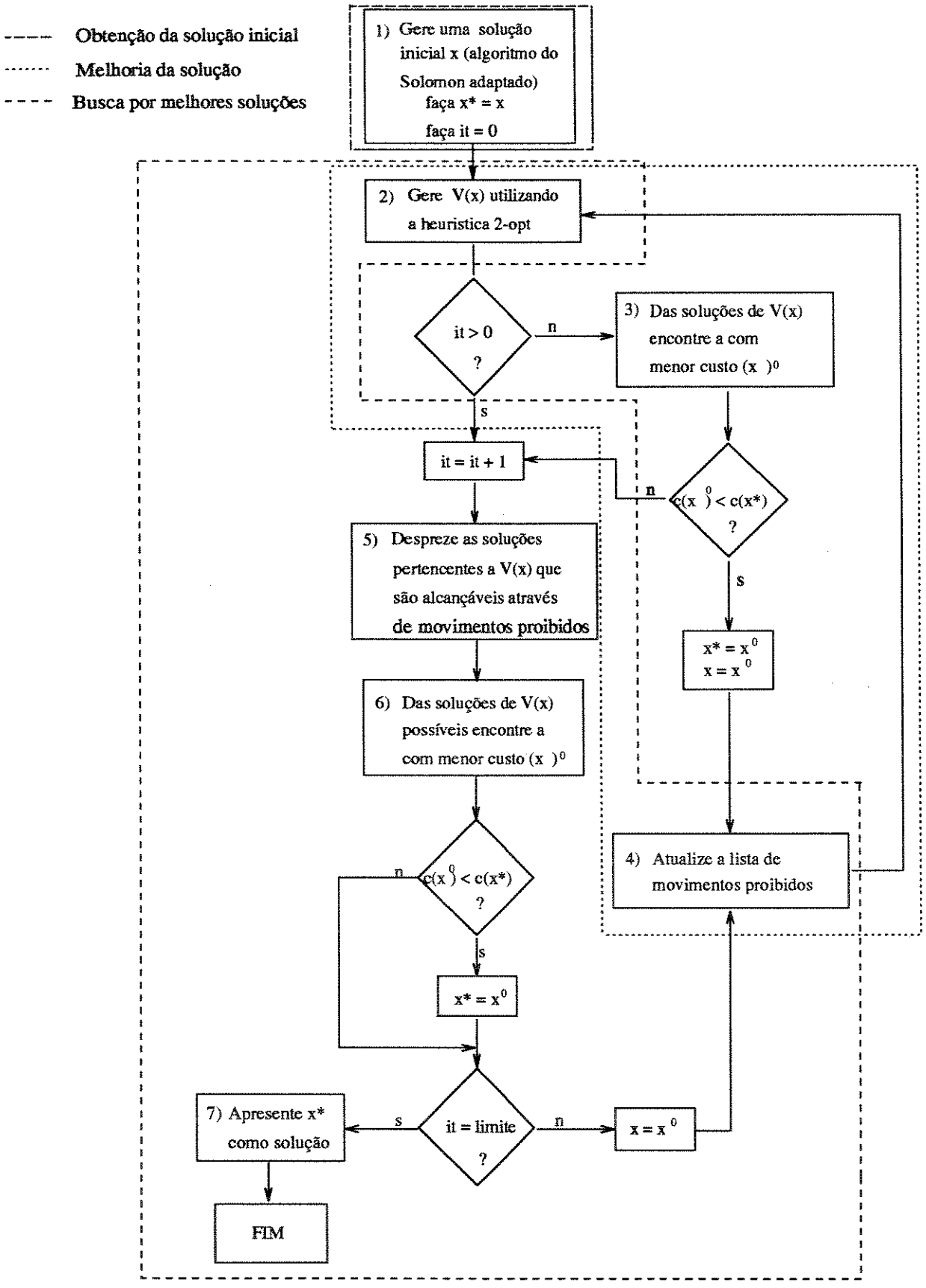

Fig. 4.1 - Heurística Proposta

4.1. Considerações Iniciais

Antes de serem apresentados os principais procedimentos da heurística, faz-se necessária a colocação de alguns aspectos práticos do problema e a consideração de algumas hipóteses que definem a estrutura do problema, passo necessário para a elaboração e entendimento da heurística.

A medida de desempenho adotada é uma função de custos causados pela preparação de máquina, atrasos de ordens de produção e adiantamentos no término de ordens, o que resulta na estocagem de produtos obtidos com o processamento dessas ordens. O objetivo na resolução do problema é encontrar uma solução, programa de produção, com o menor valor dessa função de custos.

Não se restringe a solução do problema a programas de produção sem ordens atrasadas, já que atrasos são permitidos e penalizados pela função objetivo. Não se considera também a existência de restrições tecnológicas, ou seja, qualquer sequência de ordens é possível na máquina.

Antes do processamento de qualquer ordem, a máquina se encontra em um estado de ajuste e limpeza reportado ao processamento de uma ordem fantasma, denominada ordem zero. Ao término do processamento da última ordem do programa, o estado de ajuste e limpeza da máquina deve voltar a ser o padrão representado pela ordem zero. Assim, este é um problema de n+1 ordens, onde a ordem inicial é igual a final. Como essa ordem refere-se apenas a um estado de ajuste e limpeza da máquina, não sendo responsável pelo processamento de produtos, o tempo de processamento dela é nulo.

Estando todas as ordens inclusas em qualquer programa de produção gerado, já que não havendo restrições de factibilidade por atraso e restrições tecnológicas, nenhuma ordem será deixada de fora de qualquer programa de produção, o espaço total de soluções possue (n+1)! soluções possíveis.

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

Não está se utilizando tempos de ociosidade como variáveis de decisão. Isto pois a consideração de tal variável tem implicações além do custo de estoque, tal como o aumento do *makespan*, custo de mão-de-obra parada, custo de capital fixo e outros. Isto sem contar que na cultura das empresas tais tempos não são bem aceitos, já que sempre foram associados a

ineficiência do processo ou da própria programação.

A heurística trabalha com dados determinísticos e conhecidos antes da programação. Não se consideram variações que possam ocorrer no sistema produtivo, o que resultaria em um caráter dinâmico e estocástico de dados, como tempo de processamento e tempo de preparação de máquina.

Se uma operação já foi iniciada, esta deve ser completada antes que outro processamento seja iniciado na máquina. Sendo assim, cada ordem deve ser processada por completo. Não está se considerando a possibilidade das ordens serem quebradas (*no pre-emption*).

A máquina não pode processar mais de uma ordem ao mesmo tempo. Existem equipamentos que possuem vários estágios e cada um desses estágios pode ser ajustado, ou programado, para realizar diferentes operações (dimensões, materiais, cores, ...). A heurística desenvolvida direciona-se para a programação em máquinas que processam uma ordem por vez, ou que processam produtos de uma mesma ordem em vários estágios simultaneamente.

A máquina está sempre viável, não se considera manutenção ou previsão de quebra.

Tanto os atrasos como adiantamentos das ordens de produção serão causados pela defasagem dos instantes de conclusão dessas ordens com as suas datas de entrega. Se o instante de conclusão de uma ordem for maior que a sua data de entrega, caracteriza-se um atraso. Caso contrário, caracteriza-se um adiantamento da ordem, resultando em estoque. Portanto, essa defasagem irá caracterizar dois custos, por atraso e por estoque, respectivamente. Como apresentado por Baker e Scudder (1990), vários estudos feitos em PPP mostram que a consideração do instante de início de entrega como sendo igual a data de entrega, melhor representa situações reais de fatos geradores de custos de atraso e estoque. Se a medida de desempenho adotada para a avaliação das soluções for em função apenas desses custos, a solução ideal é que cada ordem termine exatamente na sua data de entrega, ou que as defasagens sejam mínimas.

Os tempos de preparação da máquina não estão inclusos nos tempos de processamento e variam conforme a sequência das ordens. A variação desses tempos ocorre em função de uma maior ou menor dificuldade para o ajuste e limpeza da máquina ocasionada pelas ordens precedentes e sucessoras às preparações.

Muitas ordens apresentam as mesmas características de processamento, não necessitando portanto de preparação de máquina entre uma e outra. As empresas têm

desenvolvido produtos e peças que possam ser agrupados em famílias com as mesmas características de processamento, com ajustes de máquinas iguais ou muito próximos.

A heurística trabalha com tais classes de problemas, considerando tempo e custo zero para preparação de máquina entre ordens de uma mesma família.

Mediante as considerações apresentadas até aqui, pode-se representar pelo gráfico de Gantt, figura 4.2, um programa de produção resultante de tal cenário.

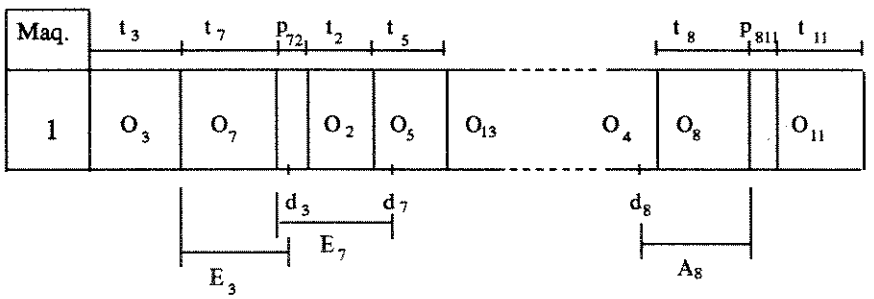

Fig. 4.2 - Gráfico de Gantt para Uma Máquina com Tempos de Preparação Dependentes da Sequência e Agrupamento em Famílias

4.2. Obtenção da Solução Inicial

A primeira etapa da heurística refere-se à construção de um programa de produção, ponto de partida para a realização das outras etapas da heurística. É sobre o programa gerado nessa etapa que são aplicados mecanismos de perturbação para geração de novos programas, vizinhos a esse, na busca por melhores soluções.

A heurística aplica sobre essa solução inicial um método de melhoria, gerando novas e melhores soluções até se atingir um ótimo local. Emprega-se então um método de busca, o qual continua gerando novas soluções, aplicando o mesmo mecanismo de perturbação usado no método de melhoria.

Portanto, o processo de busca de novas soluções se inicializa no programa gerado

nessa etapa e a sua eficiência está relacionada não apenas aos parâmetros e mecanismos que ele emprega, mas também com a região em que se encontra a solução de partida do processo de busca.

É claro que soluções iniciais geradas utilizando-se os mesmos critérios de avaliação que os critérios adotados para avaliação das soluções geradas no processo de busca, têm uma maior chance de pertencerem a uma boa região.

Como na heurística utiliza-se uma medida de desempenho ponderada, optou-se por um algoritmo de construção de programas que considerasse a ponderação desses custos para a geração da solução inicial. A heurística proposta por Solomon (1987) apresenta um bom compromisso entre qualidade de soluções e tempos computacionais.

4.2.1. Heurística de Inserção de Solomon

Proposta por Solomon para a resolução do problema de Roteamento de Veículos com Janela de tempo, a heurística constroi rotas sequenciais, onde a inicialização de cada rota é feita pela escolha de um cliente inicial usando um dos seguintes critérios:

- Cliente mais distante do nó inicial;
- Cliente com a menor data de entrega;
- Cliente com a maior quantidade a ser entregue;
- Outros.

Após iniciada a rota, o método usa dois critérios, $C_1(i, u, j)$ e $C_2(i, u, j)$, a cada iteração para inserir um novo cliente **u** na rota parcial atual, entre dois clientes adjacentes **i** e **j**.

Para cada cliente não programado em uma rota parcial $(i_0, i_1, ..., i_m)$, encontra-se o melhor lugar factível para inserção através do primeiro critério

$$C_1(i(u), u, j(u)) = \min[C_1(i_{p-1}, u, i_p)], \quad p=1,...,m \qquad (4.1)$$

A seguir o segundo critério define o melhor cliente não programado u* a ser inserido no lugar definido pelo critério 1

$$C_2(i(u*), u*, j(u*)) = melhor[C_2(i(u), u, j(u))], \qquad (4.2)$$

Quando não se pode mais encontrar clientes cuja inserção seja possível (devida a capacidade do veículo e restrições de janela de tempo ou de tempo de jornada) a heurística inicia uma nova rota, a não ser que não haja mais clientes a serem roteados.

Solomon considera, então, três enfoques que podem ser assumidos pelos critérios de escolha do local e escolha do cliente.

No primeiro enfoque o critério de escolha do local é uma ponderação entre outros dois sub-critérios, $C_{11}(i, u, j)$ e $C_{12}(i, u, j)$.

O primeiro sub-critério é expresso pela seguinte equação:

$$C_{11}(i,u,j) = d_{iu} + d_{uj} - \mu\, d_{ij} \qquad (4.3)$$

onde dij é a distância entre o cliente i e o cliente j. Se $\mu = 1$ esse critério reflete o acréscimo da distância provocado pela inserção do cliente u entre os clientes adjacentes i e j.

O segundo sub-critério é representado pela seguinte expressão:

$$C_{12}(i,u,j) = b_{uj} - b_j \qquad (4.4)$$

onde b_{uj} é o novo instante para se iniciar o serviço em j, dado que u foi inserido antes dele e b_j era o instante de início de j, antes da inserção. Sendo assim, esse sub-critério reflete o acréscimo de tempo devido a inserção do cliente u entre i e j.

O melhor lugar para inserção de um cliente não roteado é aquele que minimiza a combinação ponderada dos custos com distância e tempo para se visitar o cliente

$$C_1(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j) \qquad (4.5)$$

$$\alpha_1 + \alpha_2 = 1; \quad \alpha_1 \geq 0; \quad \alpha_2 \geq 0$$

onde $\alpha 1$ e $\alpha 2$ são pesos dos componentes dos custos considerados no critério para escolha do local de inserção.

Neste enfoque o critério para escolha do cliente a ser inserido na rota parcial é dado por:

$$C_2(i,u,j) = \lambda\, d_{0u} - C_1(i,u,j), \quad \lambda \geq 0. \qquad (4.6)$$

Se $\lambda = 1$, $C_2(i, u, j)$ verifica o benefício em se servir um cliente u através da rota parcial em

construção, ao invés dele ser atendido através de uma rota direta.

No segundo enfoque objetiva-se selecionar o cliente cujo custo de inserção minimiza uma medida ponderada da distância e do tempo totais da rota parcial. Para tal $C_1(i, u, j)$ é definido como anteriormente e $C_2(i, u, j)$ é obtido pela seguinte expressão:

$$C_2(i,u,j) = \beta_1 R_d(u) + \beta_2 R_t(u) \qquad (4.7)$$

$$\beta_1 + \beta_2 = 1, \quad \beta_1 \geq 0, \quad \beta_2 \geq 0.$$

onde $R_d(u)$ e $R_t(u)$ são a distância e o tempo totais da rota parcial atual, caso u seja inserido no lugar definido por $C_1(i, u, j)$.

No terceiro enfoque $C_{11}(i, u, j)$ e $C_{12}(i, u, j)$ são definidos como antes, contudo é acrescentado a esses a ponderação de um terceiro subcritério, representado pela seguinte expressão:

$$C_{13}(i,u,j) = l_u - b_u \qquad (4.8)$$

onde lu 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.

Nesse terceiro enfoque, Solomon emprega o critério para escolha do cliente a ser atendido como sendo igual ao critério para definição do lugar de inserção desse cliente, os quais são resultantes da expressão:

$$C(i,u,j) = \alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j) + \alpha_3 C_{13}(i,u,j) \qquad (4.9)$$

$$\alpha_1 + \alpha_2 + \alpha_3 = 1, \quad \alpha_1 \geq 0, \quad \alpha_2 \geq 0, \quad \alpha_3 \geq 0.$$

4.2.2. Adaptação da Heurística de Inserção de Solomon

A heurística apresentada por Solomon foi desenvolvida para a resolução do Problema de Roteamento de Veículos, e utiliza a construção sequencial de rotas. Essa construção pode ser feita através de três enfoques, os quais diferem na avaliação do local e do cliente a ser inserido na rota em construção.

A adaptação dessa heurística para a construção de um programa inicial de produção no problema de programação da produção com uma máquina e tempos de preparação dependentes da sequência, deve começar por restringir a construção a apenas um único programa de produção, correspondente à máquina considerada. Se houvessem restrições quanto à violação das datas de entrega e do horizonte de programação, o processo de construção do programa deveria parar quando não se encontrasse ordens de produção com inserções possíveis. Contudo, como se está considerando a penalização por violação das datas de entrega e não a restrição de tais soluções, o processo de inserção deve continuar até que não haja mais ordens que não estejam programadas.

A escolha da primeira ordem da sequência pode ser feita por critérios como:

- Ordem com a menor data de entrega;
- Ordem com o maior tempo de processamento;
- Ordem com o maior tempo de preparação em relação ao nó fantasma;
- Outros.

Como já comentado no capítulo 3, da correspondência do Problema de Roteamento de Veículos com o Problema de Programação da Produção, tem-se as seguintes equivalências de termos:

Problema de Roteamento	Problema de Programação
- Roteamento	- Programação
- Rota	- Programa de Produção
- Cliente	- Ordem de Produção
- Tempo de viagem do cliente i ao cliente j	- Tempo de preparação de Máquina entre as ordens i e j

A heurística de inserção de Solomon usa um critério para avaliação do local onde deva ser inserido o cliente e outro critério para a avaliação do cliente a ser inserido.

Como já mencionado, os critérios utilizados para a construção da solução inicial devem considerar os custos por atraso, estoque e tempo de preparação de máquina, que são ponderados na medida de desempemho a avaliar as soluções geradas no processo de busca.

O terceiro enfoque abordado por Solomon utiliza a ponderação de 3 sub-critérios para a avaliação do lugar de inserção e do cliente a ser inserido nesse, os quais possuem uma

proximidade com a medida de desempenho que irá se usar.

No primeiro sub-critério utilizado por Solomon

$$C_{11}(i,u,j) = d_{iu} + d_{uj} - \mu\, d_{ij} \qquad (4.3)$$

adotando-se $\mu = 1$, esse sub-critério avalia o acréscimo da distância na rota parcial, caso o cliente u seja inserido entre dois clientes adjacentes roteados i e j.

Se ao invés da distância entre os clientes fossem adotados os tempos de viagem entre os mesmos, que espera-se sejam proporcionais, esse sub-critério estaria avaliando o acréscimo de tempo na rota pela inserção de u entre i e j.

Com a correlação do tempo de viagem entre clientes do Problema de Roteamento de Veículos, com o tempo de preparação de máquina no Problema de Programação da Produção pode-se usar esse último na expressão 4.3 para que esse sub-critério avalie os custos com preparação de máquina na heurística de construção do programa de produção inicial. Desta forma tem-se:

$$C_{11}(i,u,j) = p_{iu} + p_{uj} - p_{ij} \qquad (4.10)$$

O segundo sub-critério apresentado

$$C_{12}(i,u,j) = b_{uj} - b_j \qquad (4.4)$$

pondera o acréscimo de tempo na rota parcial devido a inserção de um cliente u entre outros clientes i e j já roteados.

Não está se considerando de forma direta a ponderação de custos por instante de conclusão na medida de desempenho da solução, como avaliado pela expressão 4.4. Contudo, sabe-se que o acréscimo de tempo em um programa (parcial ou total) provoca alterações nas defasagens entre os instantes de conclusão e suas datas de entrega, a partir do ponto em que se deu esse acréscimo. Um acréscimo de tempo aumenta os tempos de atraso que já existem e diminui os tempos de adiantamento, ou transforma-os em atrasos.

Diminuições nos tempos de adiantamento não são abordados como custos a compor a medida de desempenho. Pelo contrário, essas são até objetivadas na programação da produção, já que esses adiantamentos é que causam custos. Porém, aumentos de tempos de atraso fomentam os custos gerados por atrasos, os quais devem ser considerados na medida de desempenho das soluções . Desta forma, o segundo sub-critério adotado deve penalizar tais custos

no processo de construção do programa pela inserção.

O atraso de uma ordem pode ser calculado pela expressão 2.4 já apresentada, contudo, expressa-se o instante de conclusão de uma ordem em função do seu início de processamento e do seu tempo de processamento. Portanto, o segundo sub-critério para a avaliação da inserção de uma ordem u entre outras duas i e j pode ser representado por:

$$C_{12}(i,u,j) = ((b_u + t_u) - d_u)^+ \qquad (4.11)$$

Essa expressão avalia portanto o atraso que incorre uma ordem u, se essa for inserida entre i e j.

O terceiro sub-critério apresentado penaliza os tempos de espera para se atender o cliente que está sendo inserido

$$C_{13}(i,u,j) = l_u - b_u \qquad (4.8)$$

Esse sub-critério correlaciona-se ao custo de estoque a ser ponderado por uma medida de desempenho objetivada. Contudo, o adiantamento em um programa de programação da produção é gerado, como visto na expressão 2.5, pela diferença entre a data de entrega e o instante de conclusão da ordem e não o seu início, já que o adiantamento na entrega de uma ordem só se caracteriza a partir da sua conclusão. Portanto, a expressão que penaliza o adiantamento de uma ordem u, se essa for inserida entre duas ordens adjacentes i e j será:

$$C_{13} = [d_u - (b_u + t_u)]^+ \qquad (4.12)$$

Nesse enfoque, 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, referentes à inserção de u entre todas as ordens i e j adjacentes já programadas e insere-se a ordem não programada entre as ordens i e j adjacentes com menor valor do critério 1. Assim

$$C_1(i,u,j) = Min(\alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j) + \alpha_3 C_{13}(i,u,j)) \qquad (4.13)$$

Solomon utiliza os parâmetros α_1, α_2 e α_3 como pesos na ponderação dos sub-critérios e estipula que $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

A adaptação desses sub-critérios foi feita visando que os mesmos expressassem custos de preparação, atraso e estoque, provocados pela inserção de uma ordem não programada

no programa parcial. Sendo assim, esses parâmetros podem ser usados para melhor expressar a influência desses custos.

O sub-critério C_{11} está analisando o aumento do tempo de preparação com a inserção de uma ordem u entre outras duas adjacentes i e j. Portanto α_1 pode ser uma medida de custo de preparação de máquina por tempo. O resultado do produto $\alpha_1 C_{11}(i, u, j)$ será o custo de preparação de máquina resultante da inserção.

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 adiantamento, respectivamente, na ordem a ser inserida. Aqui não se deve assumir que os parâmetros $\alpha 2$ e $\alpha 3$ possam refletir uma medida de custo de atraso e estoque por tempo e que o resultado dos produtos desses parâmetros pelos tempos de atraso e adiantamento resultantes dos critérios sejam os custos de atraso e estoque para essa ordem inserida, isto pois existem outros fatores que devem ser considerados para a quantificação desses custos.

As ordens podem apresentar características distintas que levam a diferentes custos de atraso ou estoque, neste caso os parâmetros $\alpha 2$ e $\alpha 3$ devem ser diferentes para cada ordem. Um dos fatores que levam a diferenças nesses custos é a quantidade de peça ou produto a ser produzida, a qual difere para cada ordem. É claro que quantidades maiores provocam custos de atraso e estoque maiores.

Outros fatores podem provocar tais diferenças, como características de estocagem e multas contratuais por atraso, os quais são diferentes para cada ordem. Contudo, esses fatores podem ou não existir em função dos produtos ou forma de trabalho das empresas.

Sendo assim, considera-se que os custos por atraso ou estoque dependem não só dos tempos de atraso e adiantamento, mas também das quantidades de produtos das ordens atrasadas ou adiantadas. Sendo q_j a quantidade de produto da ordem j, tem-se duas formas de se abordar tais custos: ou se considera parâmetros $\alpha 2$ e $\alpha 3$ diferentes para cada ordem, ou se considera que esses parâmetros reflitam o custo de atraso e estoque, respectivamente, por unidade de tempo e quantidade (R$ / min . ton, por exemplo).

Adotando-se a segunda forma de abordar esses custos, pode-se reescrever a expressão 4.13 como:

$$C_1(i,u,j) = Min(\alpha_1 C_{11}(i,u,j) + \alpha_2 C_{12}(i,u,j) q_u + \alpha_3 C_{13}(i,u,j) q_u) \quad (4.14)$$

Os critérios C_{12} e C_{13} são antagônicos, já que ambos avaliam a defasagem da data

de entrega da ordem a ser inserida e o instante de conclusão desta. Se a data de entrega for posterior ao instante de término está caracterizado que a ordem terá que esperar para ser entregue. Caso ocorra o contrário, caracteriza-se um atraso na entrega dessa ordem. Logo quando $C_{12} > 0$, C_{13} será igual a 0 e vice-versa.

Está se usando o instante de início de uma ordem mais o seu tempo de processamento para representar o instante de conclusão desta, isto pois esses instantes de início de processamento são facilmente obtidos considerando-se apenas os tempos de processamento das ordens, os tempos de preparação de máquina entre famílias e sabendo-se que o instante de início da primeira ordem no programa é igual a 0.

$$b_0 = 0$$

$$b_{i1} = b_0 + t_0 + p_{oi1} = 0$$

$$b_{i2} = b_{i1} + t_{i1} + p_{i1i2}$$

$$b_{i3} = b_{i2} + t_{i2} + p_{i2i3}$$

...

onde i1 representa a ordem na posição 1 do programa de produção, i2 a ordem na posição 2 e assim por diante.

4.3. Melhoria

A partir da solução inicial encontrada, utilizando-se a adaptação da heurística de Solomon, aplica-se um método de melhoria baseado no método de melhoria 2-opt.

Essa heurística gera a vizinhança de uma solução removendo um par de arestas não adjacentes e reconectando-se as duas cadeias resultantes de forma a constituir uma nova solução. Portanto, em um programa de n+1 ordens o m'etodo 2-opt gera um espaço de soluções de tamanho $C\frac{2}{n+1} - [(n+1) - 1]$.

Da vizinhança encontra-se o vizinho com menor custo. Se esse programa tiver um valor de custo menor que o da solução corrente, ele passa a ser a nova solução. Esse processo continua até que nenhum vizinho gerado tenha um valor da medida de desempenho melhor que o da solução que gerou a vizinhança. Essa solução é chamada então de 1º ótimo local.

O primeiro passo para a implementação desse processo de melhoria é a definição da medida de desempenho que irá avaliar os vizinhos gerados. O objetivo deste trabalho é a construção de uma heurística que considere os custos de preparação de máquina, atraso e estoque. Portanto, assim como no critério adotado para a construção do programa inicial de produção, essa medida de desempenho deve ponderar esses três custos.

A função de penalidade 2.8 apresentada no capítulo 2 deste trabalho poderia ser empregada para determinar medidas de desempenho para cada programa de produção a ser avaliado.

$$f(P) = \sum_{k=1}^{n+1} [\gamma_{j_{k-1}j_k} p_{j_{k-1}j_k} + \alpha_{j_k}(d_{j_k} - c_{j_k})^+ + \beta_{j_k}(c_{j_k} - d_{j_k})^+] \qquad (2.8)$$

Essa função apresenta pesos para os tempos de preparação, atraso e adiantamento diferentes para cada ordem, pois abrange casos em que essas penalidades variam não só pela origem (preparação, atraso ou adiantamento) mas também por aspectos específicos de cada ordem.

Está se considerando aqui, como já visto em 4.2.2., que os únicos aspectos das ordens que provocarão diferenças nos custos são as suas quantidades. Esse aspecto não irá influenciar o custo de preparação de máquina e pode ser considerado à parte nos custos de atraso e estoque de ordens. Neste caso, o peso do tempo de preparação de máquina, α_1, pode ser expresso como custo de preparação de máquina em função do tempo. E os pesos dos tempos de atraso e adiantamento das ordens em custos de atraso e estoque em função do tempo (de atraso ou adiantamento) e da quantidade de cada ordem atrasada ou estocada.

Esses custos serão os mesmos que os adotados nos critérios de inserção de 4.2.2., α_1, α_2 e α_3. Sendo assim, tem-se a função que representará a medida de desempenho para um dado programa P, M(P):

$$M(P) = \sum_{k=1}^{n+1} [\alpha_1 p_{j_{k-1}j_k} + \alpha_2 (c_{j_k} - d_{j_k})^+ q_{j_k} + \alpha_3 (d_{j_k} - c_{j_k})^+ q_{j_k}] \qquad (4.15)$$

onde k indica a posição da ordem na sequência. Logo $p_{j_{(k-1)}j_k}$ representa o tempo de preparação entre a ordem que ocupa a posição (k-1) e a ordem que ocupa a k-ésima posição.

Como dito nas considerações desse problema, não se considera tempo de preparação de máquina entre uma ordem fantasma, ordem zero, e qualquer ordem do programa,

portanto $p_{0j} = p_{j0} = 0$. E sendo a quantidade dessa ordem fantasma igual a 0 ela não irá provocar qualquer custo de programação.

Após esta etapa de melhoria a heurística passa para a etapa de Busca Tabu, portanto é interessante que em cada iteração, (movimento), dessa etapa de melhoria, se guardem os atributos caracterizadores desses movimentos na estrutura de armazenamento de atributos proibidos. Isto pois ao se sair deste primeiro ótimo local iniciando-se a busca tabu, o processo de busca se direciona para uma região diferente da já trilhada até esse ponto.

Quanto à estrutura de armazenamento de atributos, a heurística está usando duas matrizes. Uma armazena as arestas eliminadas (atributo 1) e a outra as arestas adicionadas (atributo 2) na geração de novas soluções com o 2-opt. O número de iterações em que o atributo ficará ativo é aleatório e será gerado em um intervalo determinado a partir de um estudo prévio.

Em cada movimento do método de melhoria 2-opt observa-se que após a eliminação e adição de arestas, algumas ordens trocam de lugar no programa, o que provoca a alteração dos instantes de início e término das ordens a partir da ordem predecessora à primeira aresta eliminada. Essas alterações resultam em diferentes tempos de adiantamento e atraso. Como algumas ordens mudam de lugar no programa, devido a essas alterações, tem-se também a possibilidade de alteração de alguns tempos de preparação.

A figura 4.3 ilustra as alterações decorrentes de um movimento realizado pelo método de melhoria 2-opt em um programa com 5 ordens. A ordem predecessora à primeira aresta eliminada é A. Após o movimento realizado pelo método de melhoria 2-opt observa-se que a ordem B, que tinha um tempo de adiantamento E_B, passa a ter um tempo de atraso A_B, o inverso ocorre com a ordem D (de tempo de atraso A_D passa a ter tempo de adiantamento E_D), a ordem C tem um aumento no seu tempo de atraso e a ordem E tem o seu tempo de adiantamento diminuido. Quanto aos tempos de preparação de máquina deixa de existir p_{AB} (tempo de preparação entre as ordens A e B), o tempo de preparação entre as ordens D e E não existia, pois essas ordens pertencem à mesma família, surgem os tempo de preparação p_{AD} e p_{BE}. Os tempos de preparação p_{BC} e p_{CD} passam a ser representados por p_{CB} e p_{DC}, mas continuam os mesmos, já que está se assumindo que a matriz de tempos de preparação de máquina é simétrica.

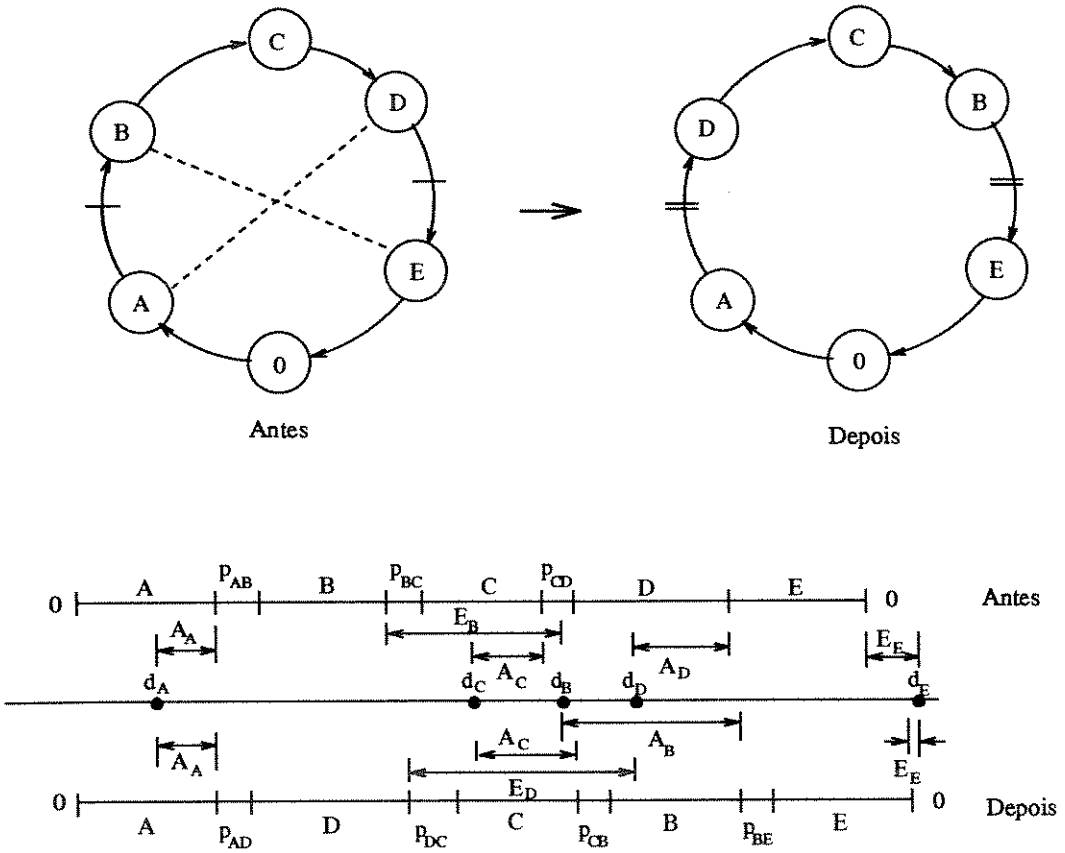

Fig. 4.3 - Alterações Decorrentes de um Movimento com o 2-opt

Um caso interessante de movimento ocorre quando a família da ordem posterior à primeira aresta eliminada é igual a família da ordem anterior a segunda aresta eliminada. Supondo o movimento em um programa de seis ordens representado pela figura 4.4, onde as ordens B e D pertencem à mesma família. Pode-se observar que não há mudança nos tempos de preparação e que o instante de conclusão da última ordem envolvida no movimento, ordem E, assim como os instantes de conclusão das ordens seguintes a essa, continuam sendo os mesmos que os instantes de conclusão dessas ordens antes do movimento. Logo, só há alteração dos tempos de atraso e adiantamento para as ordens localizadas entre a ordem anterior à primeira aresta eliminada e a ordem posterior à segunda aresta eliminada, o que pode ser verificado na figura pelas alterações nos tempos de atraso das ordens B, C e D (A_B e A_C aumentam e A_D diminui).

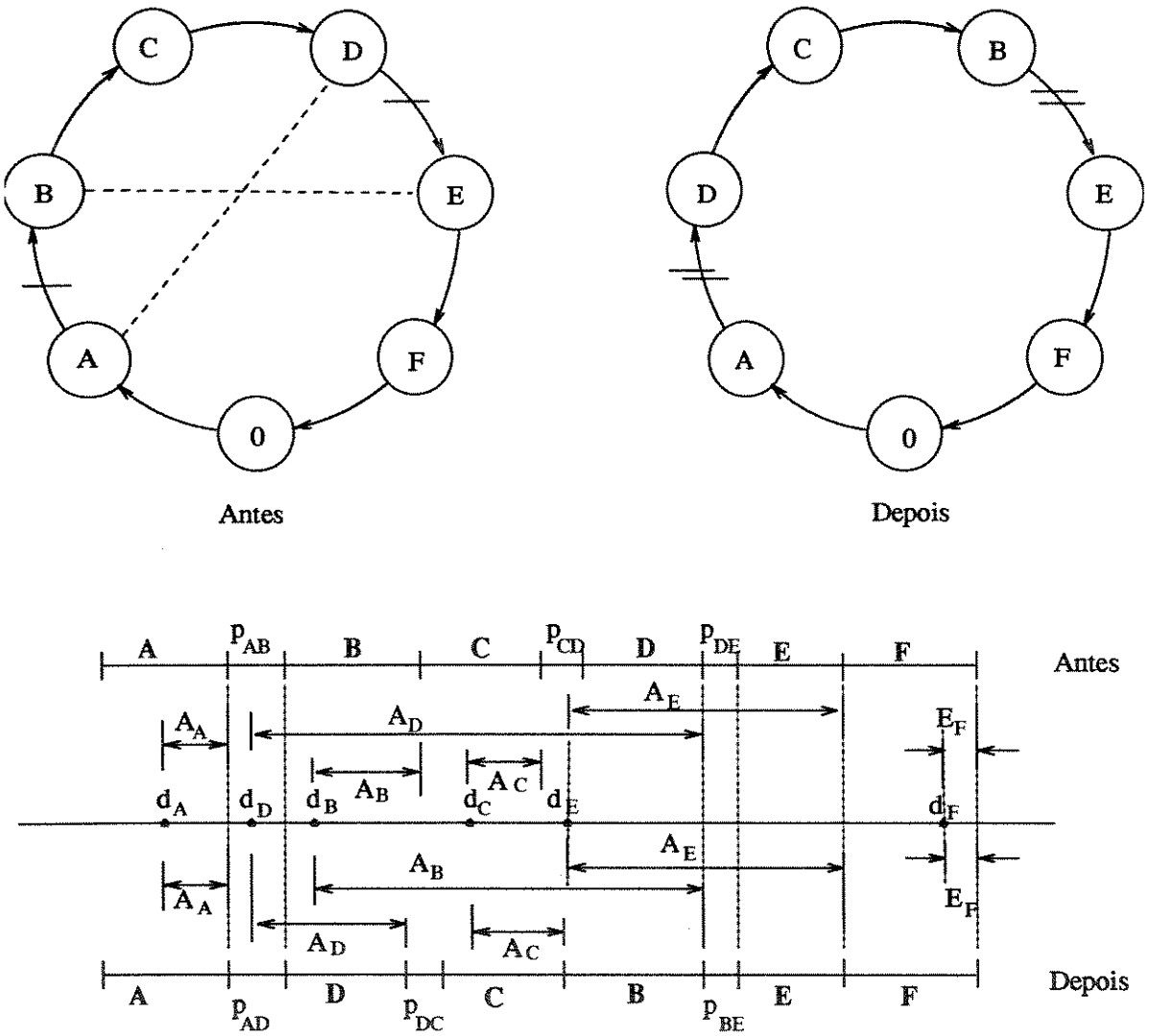

Fig. 4.4 - Movimento Sem Alteração dos Tempos de Preparação

As alterações nos tempos de atraso, adiantamento e preparação de máquina resultantes dos movimentos realizados pelo método de melhoria 2-opt, provocam também alterações nos custos gerados por esses tempos, respectivamente custo de atraso, custo de estoque e custo de preparação de máquina. Essas alterações é que geram vizinhos com diferentes custos totais, os quais são analisados para a adoção ou não de um novo programa de produção como solução atual do problema.

Deve-se lembrar que os custos de atraso e estoque de cada ordem são expressos não apenas em função dos tempos de atraso ou adiantamento da ordem, mais também em função da quantidade de produto a ser produzido por essa. Logo, pode-se ter o seguinte caso: como resultado de um movimento uma ordem A de 100 Kg de produto, que tinha um tempo de atraso de 10 horas, passou a ter um tempo de atraso de 4 horas e uma ordem B de 220 Kg de produto que tinha um tempo de atraso de 5 horas passou a ter um tempo de atraso de 9 horas. Supondo os custos de atraso (α_2) de 1 U.M.[4] / h*Kg tem-se que o tempo de atraso relativo a essas duas ordens passou de 15 hs antes do movimento para 13 horas após o mesmo, contudo essa queda no tempo de atraso não foi acompanhada pelo custo de atraso, que passou de 2100 U.M. para 2380 U.M., isto pois a queda no tempo de atraso e assim custo de atraso, da ordem A não foi sufuciente para compensar o aumento de tempo e custo de atraso da ordem B, devido a maior quantidade dessa segunda.

4.4. Busca Tabu

Com o emprego do método de busca local 2-opt, a heurística chega a uma solução que é um mínimo local, ou seja, menor valor da função de custo dentre todas as soluções vizinhas geradas. Sendo o espaço de soluções muito grande, outros mínimos locais devem existir, alguns possivelmente melhores que esse primeiro encontrado. Para se continuar a busca pelo espaço de soluções, objetivando-se encontrar mínimos locais melhores que o já obtido, há a necessidade de se usar um método de busca que ultrapasse a barreira da otimalidade local. Emprega-se para tal a **Busca Tabu**.

Como já mencionado, a Busca Tabu permite sair de um ótimo local admitindo movimentos de piora. Ela também previne ciclagens e estimula a exploração de novas regiões proibindo, por um certo número de iterações, movimentos que possam gerar soluções recentemente exploradas. Esses movimentos de piora e os movimentos sujeitos a restrições, também serão realizados com o uso do método de busca local 2-opt. O critério para avaliação

---

[4] U.M. refere-se a Unidade Monetária, geralmente a moeda corrente no pais.

das soluções geradas é o mesmo que o apresentado em 4.2, com o uso da função 4.15.

Segundo Glover (1989 e 1990), para definir movimentos com o 2-opt é necessário indicar apenas as arestas eliminadas e adicionadas nesses movimentos. Portanto, com esses atributos consegue-se definir o movimento.

Como já mencionado, a heurística está usando esses atributos para representar os movimentos. Para o armazenamento e consulta desses está-se usando duas matrizes.

As arestas a serem adicionadas em novos movimentos devem ser confrontadas com as arestas eliminadas em movimentos recentes. Se essas arestas, a serem adicionadas em novos movimentos, coincidirem com arestas recentemente retiradas de soluções e armazenadas na matriz de arestas eliminadas, pode-se estar caracterizando uma volta a essas soluções anteriores, o que não é interessante, pois o que se quer são novas soluções. Se essas coincidências possuirem uma periodicidade está ocorrendo ciclagem, o que é pior ainda, pois assim está se analisando o mesmo espaço de soluções de período em período.

Armazenando-se as arestas adicionadas em movimentos recentes, essas devem ser confrontadas com as arestas a serem eliminadas em novos movimentos. Se essas arestas, a serem eliminadas em novos movimentos, coincidirem com as arestas recentemente adicionadas de soluções e armazenadas na matriz de arestas adicionadas, pode-se estar desconsiderando a busca por uma região promissora. Região promissora, pois essas arestas adicionadas referem-se ao melhor programa de produção encontrado entre vários analisados. Mesmo sendo um movimento de piora, esse é o menos pior de todos os gerados pelo 2-opt.

Mas quando um dado movimento será considerado tabu? Quando pelo menos uma das quatro arestas que o caracterizam coincidir com uma aresta armazenada em uma das matrizes? Ou quando as quatro arestas que o caracterizam coincidir com quatro arestas armazenadas nas matrizes?

Quanto maior for o número de arestas envolvido em um movimento, que tiver que coincidir com arestas ativas nas matrizes para caracterizar a proibição desse movimento, menos restritiva será a busca e assim maiores as possibilidades de ciclagem e menos aprofundada a busca nas regiões. Por outro lado, quanto menor for este número, mais restritiva será a busca, deixando de considerar muitas soluções que poderiam direcionar essa busca para regiões mais promissoras.

A heurística trabalha com um parâmetro, **maxted**, que define tal aspecto da estratégia de busca. Maxted é o número máximo de coincidências entre arestas envolvidas nos

movimentos e arestas armazenadas nas matrizes, necessário para que um movimento não seja considerado tabu. Por exemplo, se maxted = 2 isto significa que só quando três ou quatro arestas envolvidas em um movimento coincidirem com arestas ativas nas matrizes é que o movimento será tabu. Esse procedimento pode ser representado pela figura 4.5.

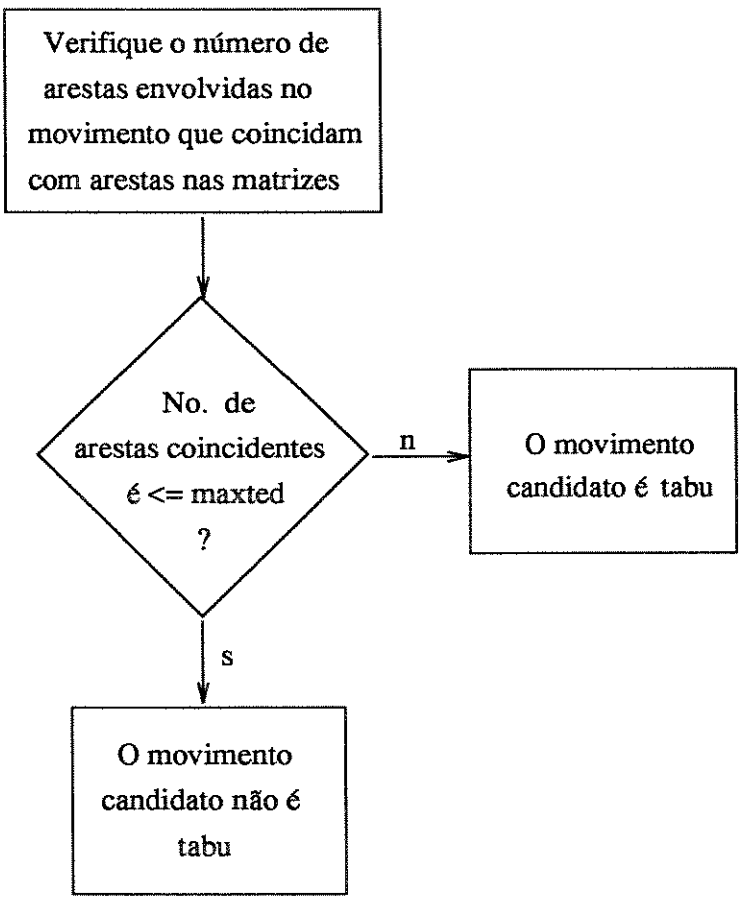

Fig. 4.5 - Caracterização de Movimento Tabu

Não apenas um movimento é proibido quando se utiliza esta estrutura, mas todos os movimentos em que o número de arestas envolvidas no movimento e coincidentes com arestas pertencentes as matrizes for maior que maxted. Logo quanto menor for maxted maior o número de soluções tabu e portanto mais restritivo é o processo de busca.

Diz-se que uma aresta está ativa ou armazenada em uma das matrizes ou pertence a uma delas, se a posição da matriz que a caracteriza possuir um número maior que o da iteração

atual.

Cada posição da matriz de arestas eliminadas representa uma aresta, isto é, esta posição é formada pela interseção de uma linha (ordem anterior) com uma coluna (ordem posterior) ordens essas que são "ligadas" por tal aresta. A matriz de arestas adicionadas é composta da mesma forma.

Para se ativar, ou armazenar, uma aresta em uma dessas matrizes, basta inserir na posicão da matriz referente à mesma, a iteração até a qual esta deve ser considerada ativa. Este número é obtido pela adição da iteração atual com o número de iterações que a aresta ficará ativa, chamado **tabu tag**.

Este procedimento de armazenamento de arestas permite a utilizacão de diferentes números de iterações em que os atributos devem ficar ativos, gerando um tamanho de lista aleatório, que como foi analisado no capítulo anterior, é mais vantajoso para se evitar ciclagens.

Além do parâmetro Maxted outro fator que restringe mais ou menos o processo de busca é o número de iterações que cada aresta fica ativa (tabu tag). Como a heurística adota um tamanho de lista aleatório, para cada aresta a ser inserida em uma das matrizes gera-se um valor de tabu tag . Esse valor será gerado aleatoriamente em um intervalo previamente definido.

Quanto maiores forem 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.

O processo de busca deve ter uma duração suficiente para que se obtenha uma boa solução. Há a necessidade portanto, de se adotar um critério de parada para o processo de busca. Na heurística é adotado um critério de parada baseado no número de iterações executados pelo processo de busca. Cada iteração refere-se à geração da vizinhança e determinação do melhor programa de produção que não tenha sido obtido com movimento tabu.

Nesse processo de controle da duração do processo de busca, a heurística usa um outro parâmetro, **limit**, para determinar o momento em que o processo de busca deve parar. Enquanto o número da iteração atual for menor que limit o processo de busca continua.

Como já foi mencionado, tanto o parâmetro maxted como o intervalo de geração dos tabu tags, influenciam o processo de busca e a qualidade das soluções encontradas. Logo, a definição desses é fundamental para se encontrar boas soluções e assim influenciar o número de iterações (limit) a serem executadas antes de se encerrar a execução da heurística.

É claro que para diferentes tamanhos de problemas (número de ordens a serem

programadas) e diferentes estruturas de problemas (número de famílias, amplitude das datas de entrega, grandeza dos tempos de processamento das ordens, etc.), os valores de maxted, intervalo de geração dos tags e limit devem ser calibrados para melhor orientar o processo de busca.

O tempo de execução da heurística também depende do tamanho do problema e, é obvio, de limit. Quanto maior for o tamanho do problema, maior o número de vizinhos a serem gerados em cada iteração, maior será também o tempo gasto para se determinar qual o vizinho que possue menor valor da função de custo.

Em um processo de busca muito restritivo pode-se ter, em uma dada iteração, todos os programas de produção que compõem a vizinhança sendo gerados por movimentos tabu. Para se continuar o processo de busca há a necessidade de se alterar tal cenário. Duas formas de se fazer tal alteração são: 1) Adota-se um outro programa de produção como sendo a solução atual do problema e a partir desse faz-se uma outra geração de vizinhança. Esta estratégia necessita da adoção de um procedimento que resulte em uma nova solução, tal como a adoção do último ótimo local como solução atual, se esse já não for a solução atual. Se for deve-se adotar um segundo procedimento. 2) Incrementa-se de um (1) o número da iteração atual, o que irá provocar a saída, ou desativação, de algumas arestas das matrizes, possibilitando assim a adoção de algum programa vizinho como solução atual do problema. Esta estratégia além de ser mais simples e fácil de implementar, condiz com a implementação simples de Busca Tabu proposta e portanto é a abordagem implementada na heurística.

Além das arestas eliminadas e das arestas adicionadas outros atributos podem ser adotados para identificar movimentos realizados pelo método de melhoria 2-opt, como por exemplo o valor da função de custo, já que a repetição desse valor indica a provável repetição de um programa de produção. Provável, pois podem existir diferentes programas de produção com um mesmo valor da função de custo, o que pode tornar a adoção desse atributo também interessante, já que não há vantagem aparente em se ficar obtendo outros programas de produção que não causem melhoria na função de custo.

Mas como diferentes programas de produção podem ter um mesmo valor da função de custos?

Em 4.3. verificou-se que quando a família da ordem posterior à primeira aresta eliminada é igual à família da ordem anterior à segunda aresta eliminada, não ocorrem alterações nos instantes de conclusão da ordem posterior à segunda aresta eliminada, assim como nas ordens seguintes a essa. Também não ocorrem alterações nos tempos de preparação, só ocorrendo

alterações nos tempos de atraso e adiantamento entre a ordem anterior à primeira aresta eliminada e a ordem posterior à segunda aresta eliminada.

Se a este caso forem adicionadas as seguintes hipóteses:

1) As defasagens entre instante de conclusão e data de entrega das ordens compreendidas entre a ordem anterior à primeira aresta eliminada e a ordem posterior à segunda aresta eliminada são do mesmo gênero (ou todas atraso ou todas adiantamento), tanto antes como depois do movimento;

2) A quantidade de produtos que são produzidas pelas ordens é proporcional ao tempo de processamento dessas ordens, e essa proporcionalidade é a mesma para qualquer ordem (proporção essa referente à velocidade de produção - Kg/min ou unidades/hs, por exemplo).

3) As ordens compreendidas entre as arestas eliminadas são da mesma família. Não há tempo de preparação entre elas.

Com essas hipóteses tem-se o caso ilustrado pela figura 4.4 que representa um movimento que obedece às hipóteses feitas. Os custos que serão alterados após o movimento são os custos das ordens B, C e D, os quais tiveram seus tempos de atraso alterados. Portanto, calculando a soma desses custos antes (1) e depois (2) do movimento, onde definiremos o atraso de uma ordem i como nA_i, tem-se:

$$(1) = \alpha_2.(q_B.A_B + q_C.A_C + q_D.A_D)$$

$$(2) = \alpha_2(q_B.nA_B + q_C.nA_C + q_D.nA_D)$$

da figura 4.4 observa-se que:

$$nA_B = A_B + t_C + t_D \quad (3)$$

$$nA_C = A_C - t_B + t_D \quad (4)$$

$$nA_D = A_D - t_B - t_C \quad (5)$$

e sendo y o fator de proporcionalidade entre a quantidade e o tempo de processamento das ordens, $q_i = yt_i$ (6). Substituindo-se (6) em (1) e (2) e substituindo-se (3), (4) e (5) em (2), tem-se:

$$(1) = \alpha_2.y.(t_B.A_B + t_C.A_C + t_D.A_D)$$

$$(2) = \alpha_2.y.[t_B.(A_B + t_C + t_D) + t_C.(A_C - t_B + t_D) + t_D.(A_D - t_B - t_C)] =$$

$$= \alpha_2.y.[t_B.A_B + t_B.t_C + t_B.t_D + t_C.A_C - t_C.t_B + t_C.t_D + t_D.A_D - t_D.t_B - t_D.t_C)] =$$

$$= \alpha_2.y.(t_B.A_B + t_C.A_C + t_D.A_D)$$

portanto, (1) = (2).

Logo, se as quantidades forem proporcionais aos tempos de processamento das ordens, se em um movimento a família da ordem posterior a primeira aresta eliminada for igual a familia da ordem posterior a segunda aresta eliminada e as defasagens entre os instantes de conclusão e as datas de entrega das ordens compreendidas entre a ordem anterior a primeira aresta eliminada e a aresta posterior a segunda aresta eliminada forem do mesmo gênero, tanto antes como após o movimento, os valores da função de custo do programa anterior ao movimento e do programa posterior ao movimento são os mesmos.

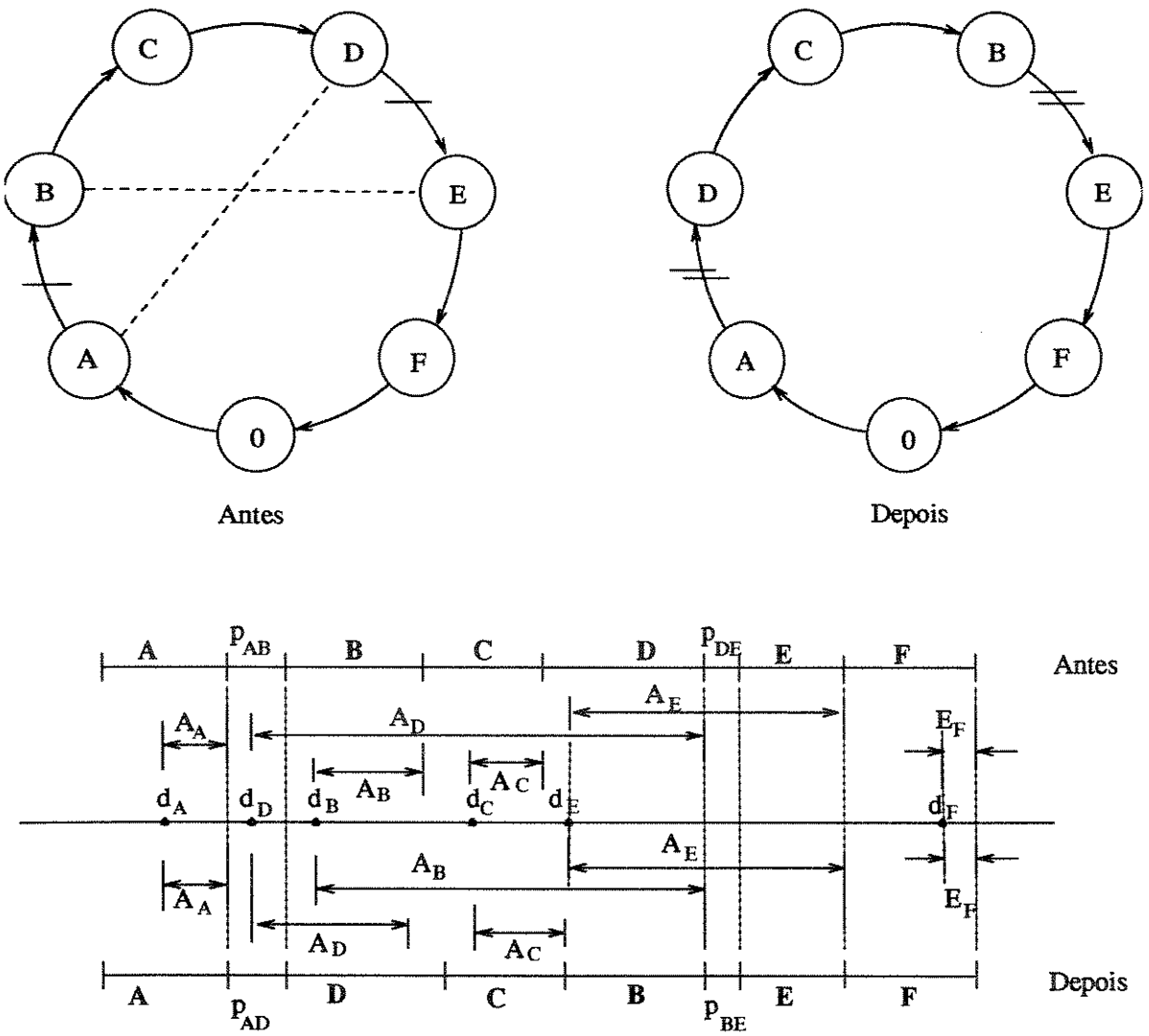

Fig. 4.6 - Movimento Sem Alteração do Custo Total

Capítulo 5

Resultados Computacionais e Conclusões

Neste capítulo são apresentados os testes computacionais realizados. O objetivo desses testes é avaliar a heurística proposta no capítulo 4, no que tange ao comportamento das suas fases durante a execução e ao comportamento da heurística frente a variações de parâmetros.

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

Os dados referentes às ordens de produção utilizados para esses testes computacionais foram gerados aleatoriamente. Cada ordem teve o seu tempo de processamento gerado aleatória e uniformemente em um intervalo, onde o limite inferior deste intervalo é tratado por **tmins** e o limite superior por **tmaxs**. As datas de entrega foram determinadas da mesma forma, onde os intervalos de geração dessas foram delimitados por uma data de entrega mínima, chamada de **datamin**, e pelo horizonte de programação, **h**.

A distribuição das ordens pelas **nf** famílias de ajustes da máquina também se deu 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 de máquina entre uma

ordem da família 2 e uma da família 3 é aproximadamente a metade do que entre uma ordem da família 1 e outra da família 3 e aproximadamente um terço do que entre ordens das famílias 1 e 4. Esta forma simplista de geração dos tempos de preparação foi adotada no início dos estudos e se baseou em problemas reais analisados na época, onde essas proporcionalidades existiam. A grandeza desses tempos de preparação são ditadas por um multiplicador, **tprep**, assim pode-se apresentar a seguinte formula geral para a geração do tempo de preparação entre duas famílias: $|f_1-f_2| * tprep * pequeno\ fator\ de\ aleatoriedade\ (\pm 0.05)$

Considera-se, nesses testes, que as quantidades de produtos a serem produzidas por cada ordem é diretamente proporcional aos tempos de processamento das mesmas e ainda que o fator de proporcionalidade, y, tem valor 1. Logo, usa-se, nos cálculos dos custos de atraso e estoque, o valor dos tempos de processamento das ordens como sendo o valor das quantidades das mesmas. Exceção faz-se à resolução de um problema real apresentado, onde se usam os dados referentes à quantidade de cada ordem.

A escolha da primeira ordem da sequência em construção, **CRITSEM**, é feita ou pelo critério da ordem com a menor data de entrega, ou pelo critério da ordem com o maior tempo de processamento.

Algumas tabelas de resultados apresentam três colunas de porcentagens (%) de melhoria, uma entre o custo do primeiro programa (solução de partida), obtido com a adaptação da heurística de Solomon, e o custo do programa referente ao primeiro ótimo local encontrado, a segunda entre o custo do programa referente ao primeiro ótimo local e o custo da melhor solução encontrada com o uso de Busca Tabu, e por último, entre o custo do primeiro programa e o custo da melhor solução encontrada com o uso de Busca Tabu.

5.1. Análises Preliminares

A eficiência da heurística na obtenção de bons resultados depende principalmente das definições dos valores de maxted e do intervalo de geração dos tabu tags. Portanto, a determinação prévia desses parâmetros é fundamental para o estudo de vários aspectos relacionados com o desempenho da heurística.

É interessante verificar também o comportamento do processo de resolução de

problemas genéricos de programação da produção em uma máquina com tempos de processamento dependentes da sequência, já que, segundo Pureza (1990), a aplicação de técnicas tabu resulta em comportamentos de busca típicos, que podem ser percebidos pela análise da progressão do processo iterativo.

Um outro aspecto importante a ser verificado no emprego da heurística, é o tempo de execução na resolução de problemas com diferentes tamanhos. Para se ter uma noção de tal aspecto, apurou-se então o tempo de execução para a resolução de quatro problemas de tamanhos diferentes, cada qual com três diferentes limites de encerramento do processo de busca. Tal estudo está apresentado na tabela 5.1.

Não houve preocupação, neste primeiro estudo, com os resultados em si, mas apenas com o tempo gasto para a execução de problemas com diferentes tamanhos. Adotou-se portanto um valor "intermediário" do parâmetro maxted para a resolução dos problemas, maxted = 2. O valor do intervalo de geração dos tabu tags não deve influenciar os tempos de execução, já que o processo de consulta aos atributos armazenados em matrizes, independe dos valores dos tabu tags.

n	limit	tempo (s)
20	100	6
	250	14
	500	27
50	100	78
	250	170
	500	338
75	100	265
	250	586
	500	1117
100	100	725
	250	1487
	500	2769

Tabela 5.1 - Tempos de Execução em Função de **n** e **limit**

Verifica-se que o tempo de execução do problema cresce de forma exponencial com o número de ordens a serem programadas, o que pode ser visualizado na figura 5.1 a seguir. Isto se deve ao crescimento exponencial do número de vizinhos a serem gerados em cada iteração com o método 2-opt.

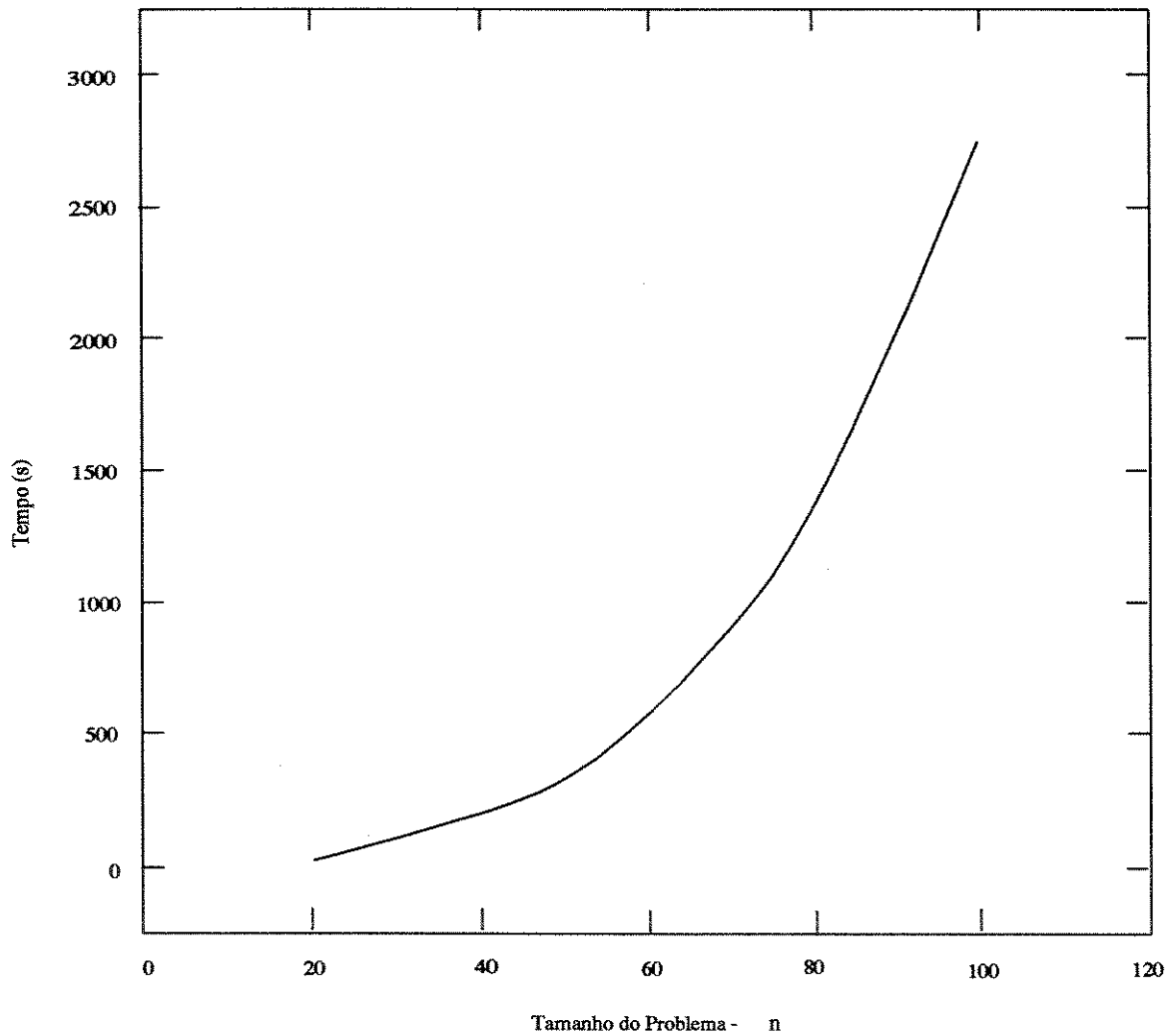

Fig. 5.1 - Tempo de Execução X Tamanho de Problema
(limit = 500)

5.1.1. Comportamento da Heurística

Antes de se partir para qualquer análise mais específica da heurística, faz-se aqui um estudo sobre o comportamento do processo de busca resultante da aplicação da heurística na resolução de um PPPUM com tempos de preparação dependentes da sequência. Para tal segue-se uma descrição detalhada do processo de busca sustentada pelo gráfico de Custo de Solução X Iteração, figura 5.2. Nesse gráfico, assim como em todos os outros desse tipo que são apresentados, as iterações não se referem somente ao processo de busca, mas também a solução de partida e aos movimentos de melhoria realizados até à obtenção do primeiro ótimo local.

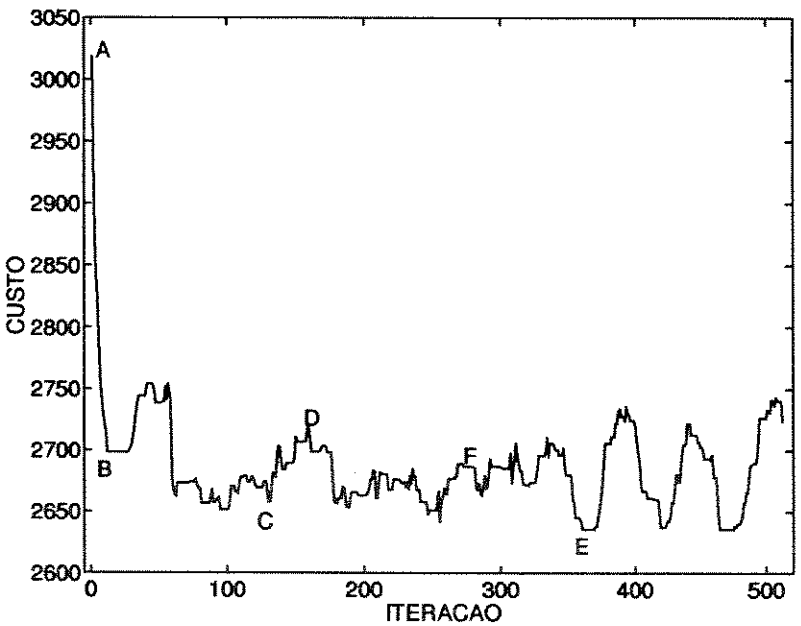

Fig. 5.2 - Comportamento Geral do Processo de Busca

Após a obtenção da solução de partida (ponto A), inicia-se o processo de melhoria. No início os custos decaem muito e rapidamente até se encontrar o primeiro ótimo local (ponto B). Esse ponto determina o início da fase tabu, marcado pela ocorrência de um pequeno platô (sequência de soluções que possuem um mesmo custo). Isto se deve ao fato do número de

iterações do processo de melhoria ter sido pequeno, dez iterações, e assim se ter poucas arestas ativas nas matrizes. Quando o número de arestas ativas nas matrizes é suficiente para evitar soluções com um mesmo custo, verifica-se um período com tendência ao aumento do custo da solução, ao qual segue uma oscilação de resultados, apresentando períodos de melhora até um outro ótimo local, como o ponto C, e períodos de piora até picos (ponto D, por exemplo). Por volta da iteração 350 a busca encontra uma região de boas soluções e inicia um processo contínuo de reduções de custo, até chegar ao ponto E, melhor minímo local até o momento e que não será superado nas iterações seguintes. Durante o processo de busca podem ocorrer outros platôs, como na região F, onde a duração e frequência desses está correlacionada aos parâmetros maxted e tabu tag a serem utilizados pela heurística.

Os valores dos parâmetros podem fazer também com que a busca se torne muito restritiva, aumentando bastante a amplitude das oscilações do processo de busca, ou podem fazer com que essa busca se torne pouco restrita, aumentando a ocorrência de platôs ou provocando a ocorrência de ciclagens, como no processo representado pela figura 5.3.

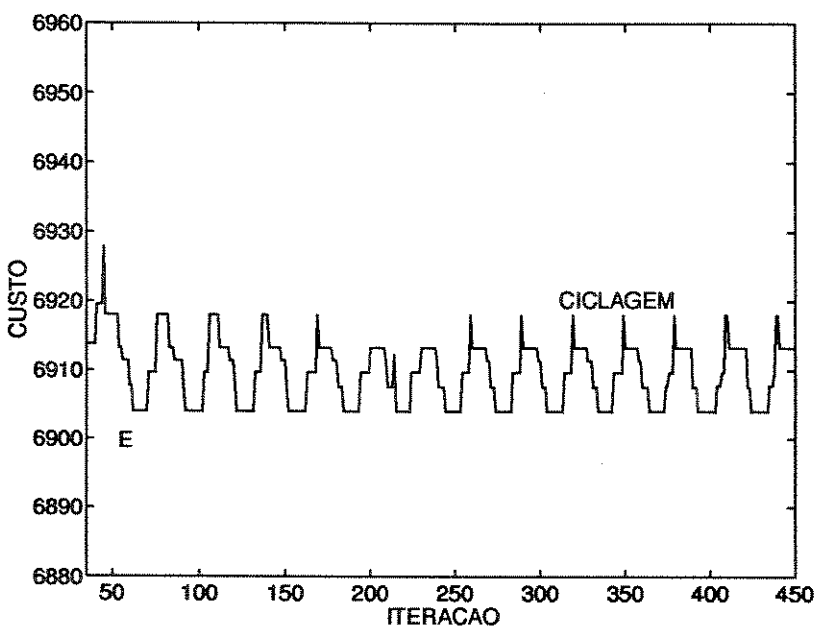

Fig. 5.3 - Ciclagem

5.1.2. Calibragem de Parâmetros

Os parâmetros que ditam a eficiência da heurística, maxted, intervalo de geração dos tabu tags e limit, podem assumir um número infinito de valores, porém, existe um *mix* de valores, *ou mix* de faixas de valores desses parâmetros em que ocorrem melhores soluções.

Como já foi mostrado, o parâmetro maxted é o número máximo de coincidências entre as arestas envolvidas em um movimento e as arestas armazenadas nas matrizes, necessário para que o movimento não seja considerado tabu. Logo, se maxted=0 isto implica que se pelo menos uma aresta envolvida no movimento estiver ativa em uma das matrizes, o movimento já será considerado tabu; se maxted=1 implica que pelo menos duas arestas envolvidas no movimento têm que estar ativas em uma das matrizes para que o movimento seja considerado tabu. Esse mesmo procedimento deve ser aplicado para maxted=2 e maxted=3. Se maxted=4 o movimento só será considerado tabu se mais de 4 arestas envolvidas no mesmo estiverem ativas nas matrizes, contudo, sabe-se que um movimento realizado com o 2-opt, em um programa de produção para uma máquina, envolve quatro arestas, logo, se maxted $\geq$ 4 nunca se terá um movimento tabu no processo de busca.

Sendo assim, avaliou-se o comportamento do processo de busca para o parâmetro maxted assumindo os valores 0, 1, 2 e 3, em um problema com 20 ordens e em um problema com 50 ordens de produção. Para cada um desses problemas os parâmetros limit e intervalo de geração dos tabu tags assumem diferentes valores, formando diferentes casos, como mostrado nas tabelas 5.2 e 5.3. Os melhores resultados na resolução dos problemas com um mesmo limit estão destacados nas tabelas.

Na análise dessas tabelas observa-se uma diferença no comportamento do processo de busca em função do tamanho dos problemas, o que já era de se esperar, pois quanto maior o número de ordens a serem programadas maior o espaço de busca e as possibilidades de vizinhos a serem visitados. Portanto, as melhores soluções para o problema menor, 20 ordens, foram encontradas com o uso de maxted = 3, tanto para limit = 150 quanto para limit =300. Para o problema com 50 ordens com limit = 200 a melhor solução encontrada corresponde a maxted = 2 e com limit = 400 a melhor solução corresponde a maxted = 1.

Em cada iteração do processo de busca escolhe-se o melhor vizinho e as arestas envolvidas na obtenção desse são armazenadas nas matrizes. Quanto menor maxted, menor o número de arestas envolvidas em um movimento que têm que estar ativas para que o movimento

seja considerado tabu, isto faz o processo de busca explorar cada vez mais soluções que não contenham arestas que caracterizem boas soluções.

Assim, quanto menor o problema mais relaxada deve ser a busca, pois senão não sobram muitos vizinhos para se seguir com a busca, e os vizinhos que sobram têm uma menor probabilidade de possuirem características de boas soluções.

Dados Comuns: n = 20; h. = 700; tmaxs = 55; tmins = 5; tprep = 3; nf = 4; critsem = < data de entr.; datamin = 50; $\alpha_1 = 3$; $\alpha_2 = 0.01$; $\alpha_3 = 0.01$

limit.	maxted	Caso	tabu tag elim/ad	tempo (s)	Custo da Solução	Iteração
150	0	1	15-25/15-25	8	458.52	0
		2	20-30/20-30	8	458.52	0
	1	3	15-25/15-25	8	455.42	6
		4	20-30/20-30	8	455.42	6
	2	5	15-25/15-25	8	447.72	26
		6	20-30/20-30	8	448.43	114
	3	7	15-25/15-25	8	447.47	48
		8	20-30/20-30	8	437.74	51
300	0	9	15-25/15-25	15	458.52	0
		10	20-30/20-30	15	458.52	0
	1	11	15-25/15-25	15	455.42	6
		12	20-30/20-30	15	455.42	6
	2	13	15-25/15-25	16	443.10	298
		14	20-30/20-30	15	448.13	300
	3	15	15-25/15-25	15	442.47	298
		16	20-30/20-30	16	437.74	51

Tabela 5.2 - Análise do Parâmetro **maxted**

(n = 20)

Dados Comuns: n = 50; h. = 1800; tmaxs = 55; tmins = 5; tprep = 5; nf = 7; critsem = < data de entr.; datamin. = 150; $\alpha_1 = 3$; $\alpha_2 = 0.01$; $\alpha_3 = 0.01$

limit.	maxted	Caso	tabu tag elim/ad	tempo (s)	Custo da Solução	Iteração
200	0	17	20-30/20-30	136	2363.81	0
		18	25-40/25-40	136	2363.81	0
	1	19	20-30/20-30	136	2328.98	14
		20	25-40/25-40	138	2328.98	14
	2	21	20-30/20-30	139	2292.87	71
		22	25-40/25-40	138	2315.93	9
	3	23	20-30/20-30	138	2298.89	36
		24	25-40/25-40	140	2293.11	110
400	0	25	20-30/20-30	263	2363.81	0
		26	25-40/25-40	264	2363.81	0
	1	27	20-30/20-30	263	2328.98	14
		28	25-40/25-40	264	2257.74	372
	2	29	20-30/20-30	272	2290.79	285
		30	25-40/25-40	271	2315.93	9
	3	31	20-30/20-30	274	2298.89	36
		32	25-40/25-40	279	2290.79	222

Tabela 5.3 - Análise do Parâmetro **maxted**

(n = 50)

Quanto aos tempos de execução, quando n = 20 esses são muito baixos para se perceber qualquer variação, já para n = 50 percebe-se que quanto maior maxted maior o tempo gasto na execução da heurística, isto pois em cada iteração gera-se toda a vizinhança da solução corrente, desprezando-se os vizinhos obtidos com movimentos tabus. Passa-se então, para a rotina de obtenção do melhor vizinho gerado. Quanto mais restritiva a busca, maior o número de vizinhos gerados por movimentos tabus, logo, menor o número de soluções a serem ordenadas

para a obtenção do melhor vizinho e assim menor o tempo gasto nessa rotina.

Para ilustrar o comportamento do processo de busca em função dos valores de maxted, apresentam-se a seguir cinco gráficos. Cada gráfico apresenta a variação dos custos das soluções na resolução de um mesmo problema, empregando-se para tal diferentes valores de maxted. Os quatro primeiros gráficos foram obtidos do processo de solução para os casos 17, 19, 21 e 23. O último gráfico foi obtido, empregando-se para o mesmo problema, maxted = 4, objetivando-se ilustrar um processo de solução quando nenhum movimento é tabu.

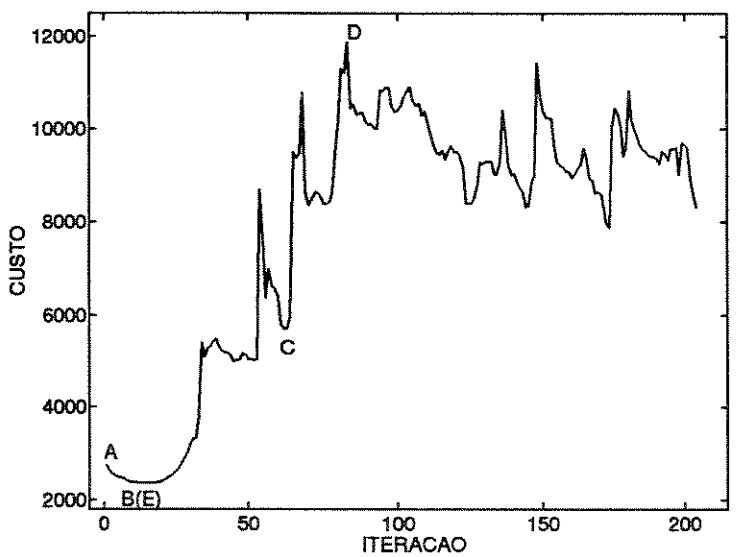

Fig. 5.4 - Processo de Resolução (maxted = 0)

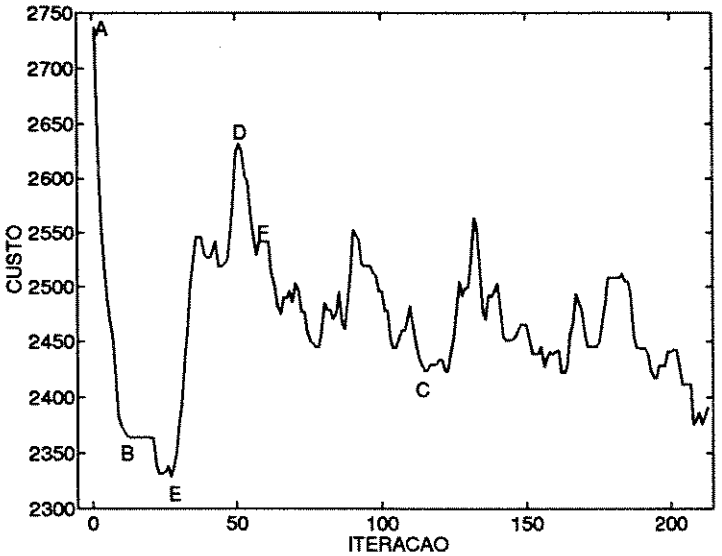

Fig. 5.5 - Processo de Resolução (maxted = 1)

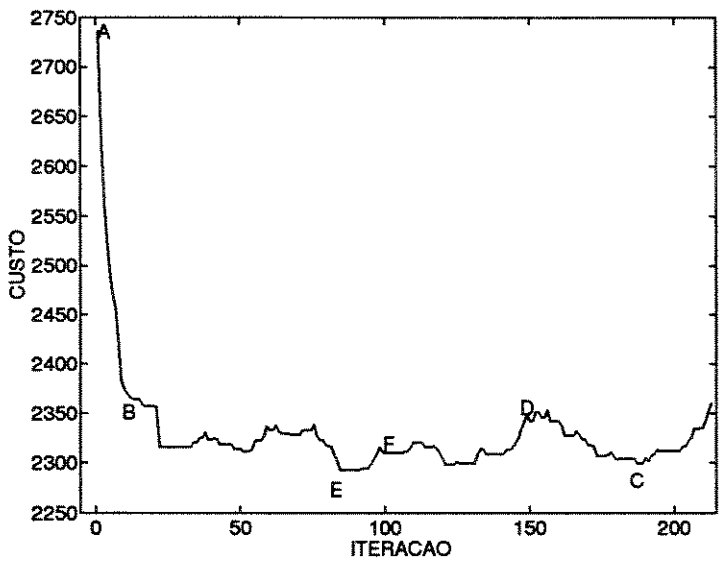

Fig. 5.6 - Processo de Resolução (maxted = 2)

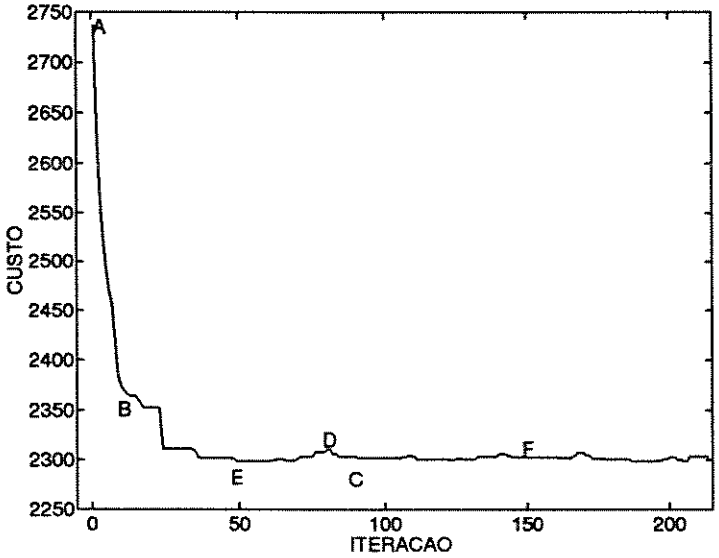

Fig. 5.7 - Processo de Resolução (maxted = 3)

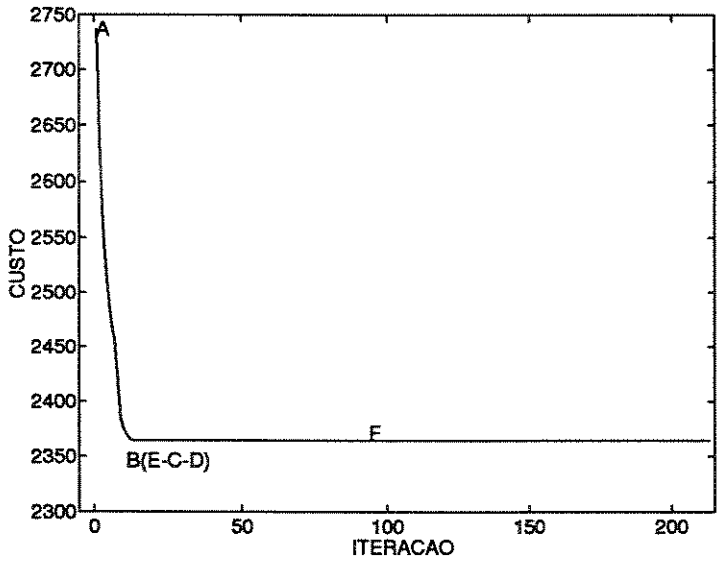

Fig. 5.8 - Processo de Resolução (maxted = 4)

A figura 5.4 representa um processo de busca muito restritivo, maxted = 0. Como mencionado, tal processo de busca deixa de explorar soluções de boa qualidade. Pela figura pode-se observar tal comportamento. Após atingir o primeiro ótimo local e iniciar o processo de busca tabu, começa a se realizar movimentos que não utilizam arestas recentemente eliminandas, que estavam se mantendo nos melhores vizinhos gerados, adicionando-se arestas que não vinham sendo adicionadas na obtenção desses melhores vizinhos.

Conforme vai se aumentando o valor de maxted o processo de busca torna-se menos restritivo e as soluções encontradas em cada iteração contêm mais características das soluções anteriores recentemente exploradas, o que provoca uma menor variação dos custos dessas soluções aumentando também a ocorrência de platôs.

Para maxted = 1, figura 5.5, a busca ainda é bem restritiva fazendo com que ainda se percam muitas características de boas soluções. Contudo as soluções exploradas já não são tão ruins quanto para maxted = 0. Alias a pior solução encontrada no processo de busca (D) tem um custo menor que a solução de partida (A). Essa busca, ainda bastante restritiva, faz com que o processo de busca mantenha uma grande variação entre as soluções, implicando em uma maior probabilidade de se encontrar uma solução com características distintas do primeiro ótimo local e com um custo bem menor que o deste, maior diversificação (o que acaba ocorrendo quando limit = 400).

Quando maxted = 2, figura 5.6, observa-se uma diminuição da variação dos custos das soluções encontradas no processo de busca, todas as soluções encontradas possuem custos menores do que o primeiro ótimo local encontrado, contudo, essa menor variação implica em uma menor diversificação e por consequência a possibilidade de se encontrar uma solução muito boa em uma região muito distinta do primeiro ótimo local é pequena, o que se torna ainda mais crítico quando maxted = 3, figura 5.7.

Quando não se possue movimentos tabu, maxted $\geq$ 4 (figura 5.8), o processo de busca restringe-se a soluções que possuam o mesmo valor de custo, pois, como visto no capítulo 4, pode ocorrer de várias soluções possuirem um mesmo valor da função de custos. Logo entre um movimento que implique em piora de custo e um que mantenha este, a heurística opta pelo segundo, produzindo um platô imenso, talves perpetuo, pois pode-se ficar repetindo algumas soluções indefinidamente.

A avaliação do intervalo de geração dos tabu tags e limit foi feita testando-se várias combinações desses parâmetros em problemas com 20 ordens de produção e em problemas

com 50 ordens. Para cada combinação de limit e intervalo de geração dos tabu tags foram gerados 10 problemas com diferentes sementes. As tabelas 5.4 e 5.5 a seguir apresentam as porcentagens de melhoria (média de cada conjunto de 10 problemas) para diferentes combinações de limit e intervalos de geração dos tabu tags, intervalos esses correspondentes aos tabu tags para arestas eliminadas (elim) e para arestas adicionadas (ad). Quanto maior a porcentagem de melhoria para problemas com o mesmo tamanho, mais eficiente foi o processo de busca, alcançando melhores resultados finais, já que as soluções obtidas até o primeiro ótimo local são as mesmas, pois os dados de entrada são os mesmos .

Deve-se observar que um movimento em um problema pequeno causa uma alteração na sequência de uma das suas soluções bem maior, proporcionalmente, que um movimento em um problema grande. Portanto, quanto maior o problema, maiores devem ser os valores de tabu tag, isto para que a proibição de arestas envolvidas em soluções recentes, consiga forçar o mecanismo de busca a gerar soluções diferentes das recentemente exploradas.

Dados Comuns: tmaxs:80, tmins:10, tprep.:5, datamin.:300, critsem: <data de entr., α_1:30, α_2:0.1, α_3:0.1, maxted= 2

Caso	Tabu Tag		% de Melhoria		
	Elim	Ad	Sol-1o.O.L.	1o.O.L.-B.T.	Sol-B.T
33 n: 20 h: 900 nf: 4 limit: 100	0 - 10	0 - 10	9.5	0.9	10.3
	5 - 15	5 - 15	9.5	1.4	10.8
	15 - 25	15 - 25	9.5	2.1	11.4
	25 - 35	25 - 35	9.5	1.1	10.5
	25 - 35	5 - 15	9.5	1.4	10.8
	5 - 15	25 - 35	9.5	1.4	10.8
34 n: 20 h: 900 nf: 4 limit: 250	0 - 10	0 - 10	9.5	0.9	10.3
	5 - 15	5 - 15	9.5	1.5	10.9
	15 - 25	15 - 25	9.5	2.3	11.5
	25 - 35	25 - 35	9.5	1.3	10.6
	25 - 35	5 - 15	9.5	1.6	11.0
	5 - 15	25 - 35	9.5	2.0	11.3
35 n: 20 h: 900 nf: 4 limit: 500	0 - 10	0 - 10	9.5	0.9	10.3
	5 - 15	5 - 15	9.5	1.5	10.9
	15 - 25	15 - 25	9.5	2.7	11.9
	25 - 35	25 - 35	9.5	1.9	11.2
	25 - 35	5 - 15	9.5	1.6	11.0
	5 - 15	25 - 35	9.5	2.3	11.6

Tabela 5.4 - Análise do Parâmetro Intervalo de Geração dos Tabu Tags

(n = 20)

Dados Comuns: tmaxs:80, tmins:10, tprep.:5, datamin.:300, critsem: < data de entr., α_1:30, α_2:0.1, α_3:0.1, maxted = 2

Caso	Tabu Tag		% de Melhoria		
	Elim	Ad	Sol-1o.O.L.	1o.O.L.-B.T.	Sol-B.T
36 n: 50 h: 2500 nf: 10 limit: 100	5 - 15	5 - 15	17.7	0.2	17.9
	15 - 25	15 - 25	17.7	0.9	18.5
	20 - 35	20 - 35	17.7	1.7	19.1
	25 - 40	25 - 40	17.7	1.9	19.3
	40 - 50	40 - 50	17.7	1.8	19.2
	15 - 25	35 - 45	17.7	1.0	18.5
37 n: 50 h: 2500 nf: 10 limit: 250	5 - 15	5 - 15	17.7	0.2	17.9
	15 - 25	15 - 25	17.7	1.2	18.7
	20 - 35	20 - 35	17.7	2.7	19.9
	25 - 40	25 - 40	17.7	2.7	19.9
	40 - 50	40 - 50	17.7	3.5	20.6
	15 - 25	35 - 45	17.7	2.5	19.8
38 n: 50 h: 2500 nf: 10 limit: 500	5 - 15	5 - 15	17.7	0.3	17.9
	15 - 25	15 - 25	17.7	1.2	18.7
	20 - 35	20 - 35	17.7	3.2	20.3
	25 - 40	25 - 40	17.7	4.8	21.7
	40 - 50	40 - 50	17.7	3.7	20.8
	15 - 25	35 - 45	17.7	3.2	20.3

Tabela 5.5 - Análise do Parâmetro Intervalo de Geração dos Tabu Tags

(n = 50)

Analisando-se as tabelas, confirma-se a observação já feita, de que o intervalo de geração dos tabu tags são proporcionais ao tamanho dos problemas, pois quando n = 20 os melhores intervalos de geração dos tabu tags foram de 15-25, tanto para arestas eliminadas quanto para arestas adicionadas e quando n = 50 esses intervalos foram de 25-40 para limit = 100 e para limit = 500 e de 40-50 para limit = 250, também para as duas matrizes.

Outra observação que pode ser feita das tabelas é que tanto para n = 20 quanto para n = 50 as soluções obtidas utilizando-se limit = 500 são melhores que para limit = 100 e limit = 250, e que essa melhoria é mais acentuada para as faixas de valores de tabu tags que melhores resultados produziram. Isto quer dizer que o comportamento do processo de busca, para os valores de geração dos tabu tags que resultam em menores custos, apresenta uma variação tal que permite avaliar com uma certa profundidade diferentes regiões, possibilitando encontrar boas soluções dessas regiões, algumas melhores que as já encontradas. Esse processo de sucessivas melhorias pode se arrastar por muitas iterações e a definição de limit deve ser feita considerando-se a possibilidade de tais melhorias e o tempo e custo computacionais necessários para a obtenção das mesmas.

Para se verificar então o comportamento do processo de Busca em função dos intervalos de geração dos tabu tags, aplicou-se a heurística na resolução de um problema com 50 ordens, empregando-se para tal diferentes valores de intervalos de geração de tabu tags. As variações de custo no decorrer do processo de resolução para cada conjunto de intervalo de geração dos tabu tags, encontram-se ilustradas nas figuras 5.9 a 5.13.

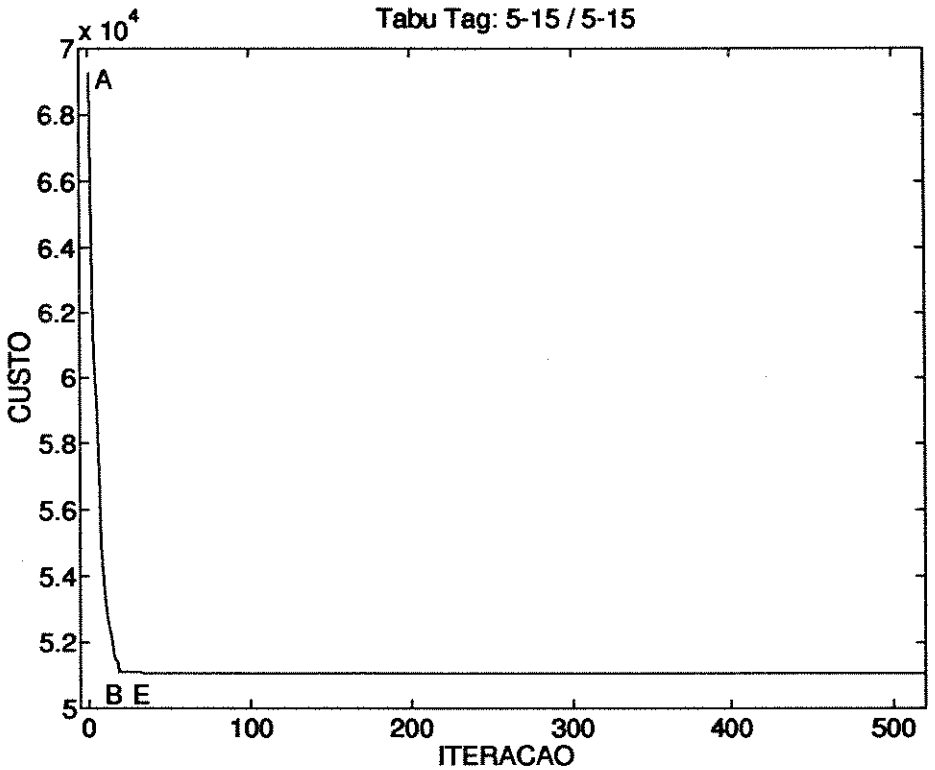

Fig. 5.9 - Análise de Intervalos de Geração dos Tabu Tags

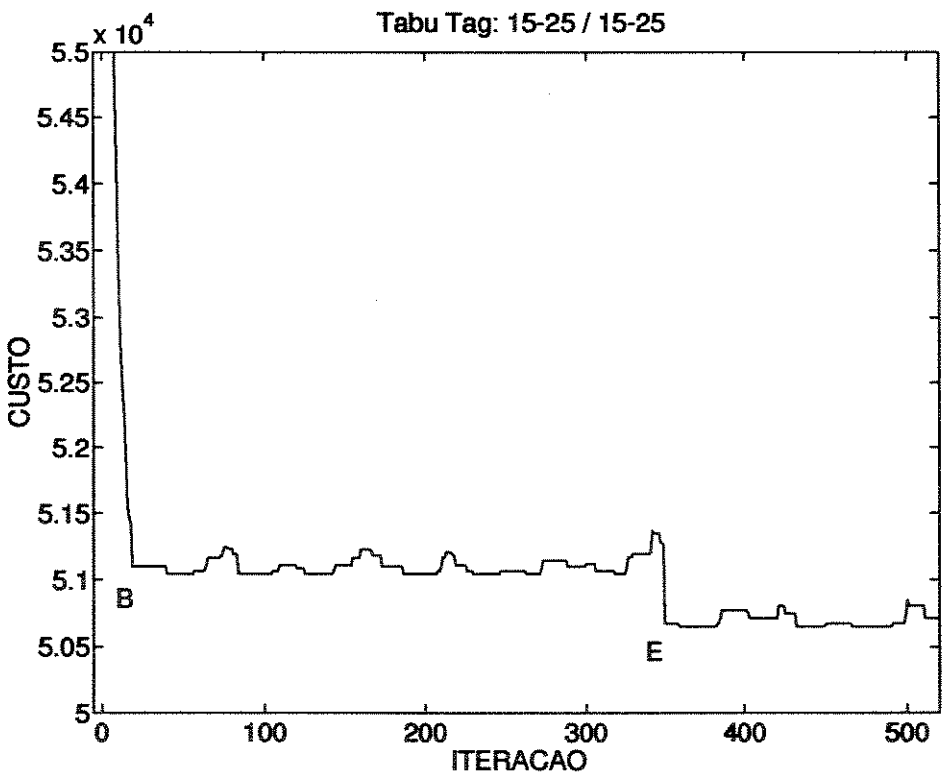

Fig 5.10 - Análise de Intervalos de Geração dos Tabu tags

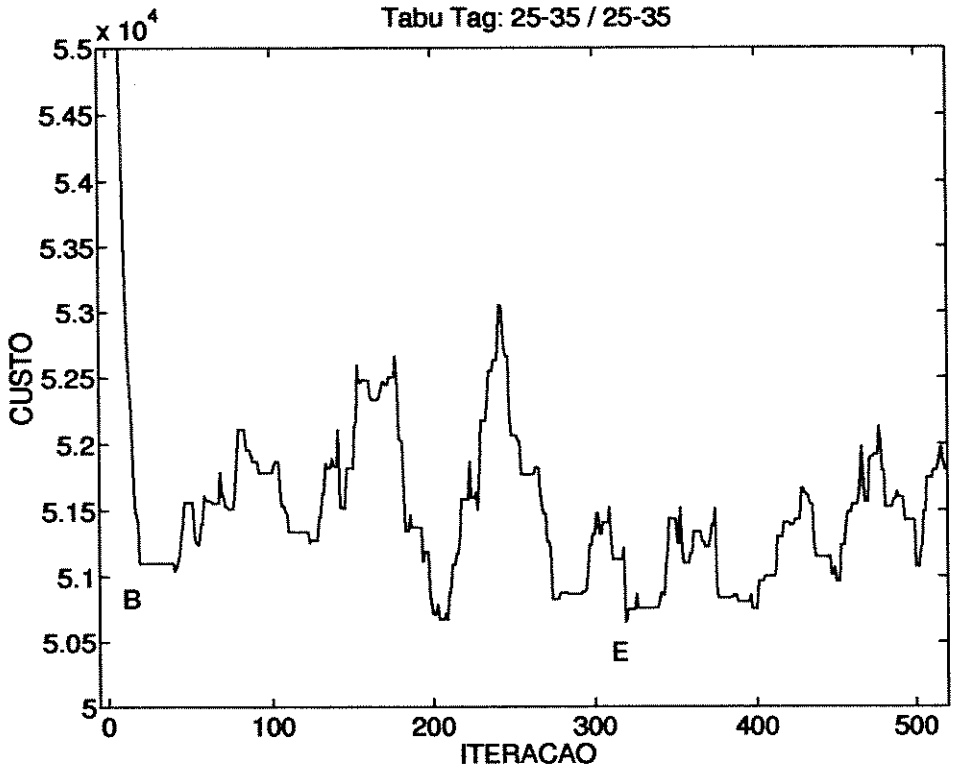

Fig 5.11 - Análise de Intervalos de Geração dos Tabu tags

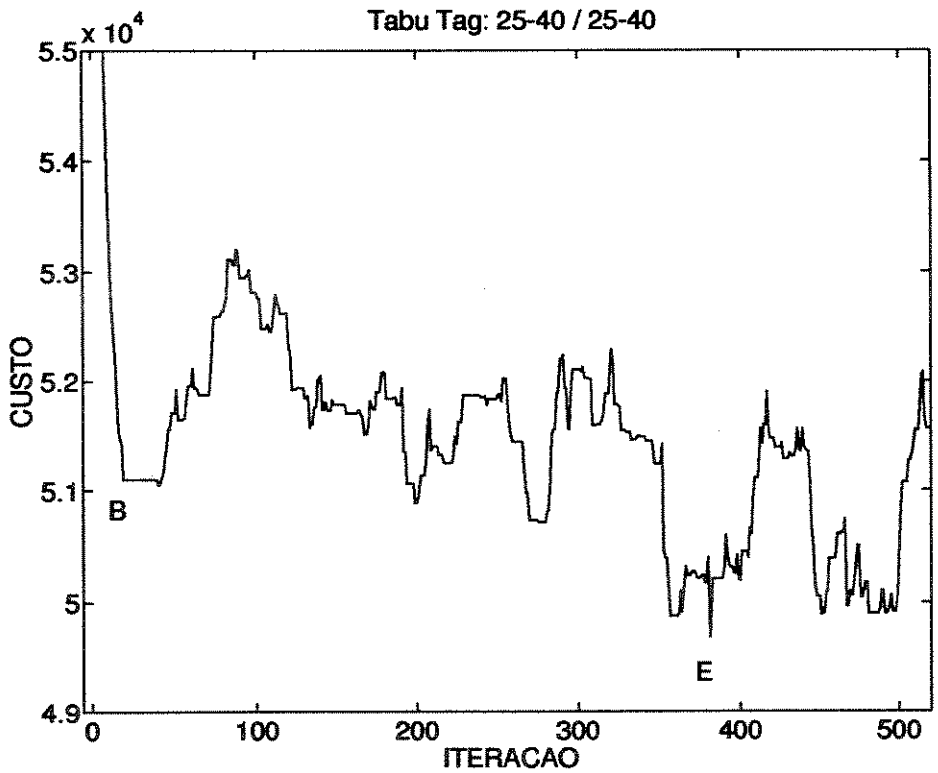

Fig 5.12 - Análise de Intervalos de Geração dos Tabu tags

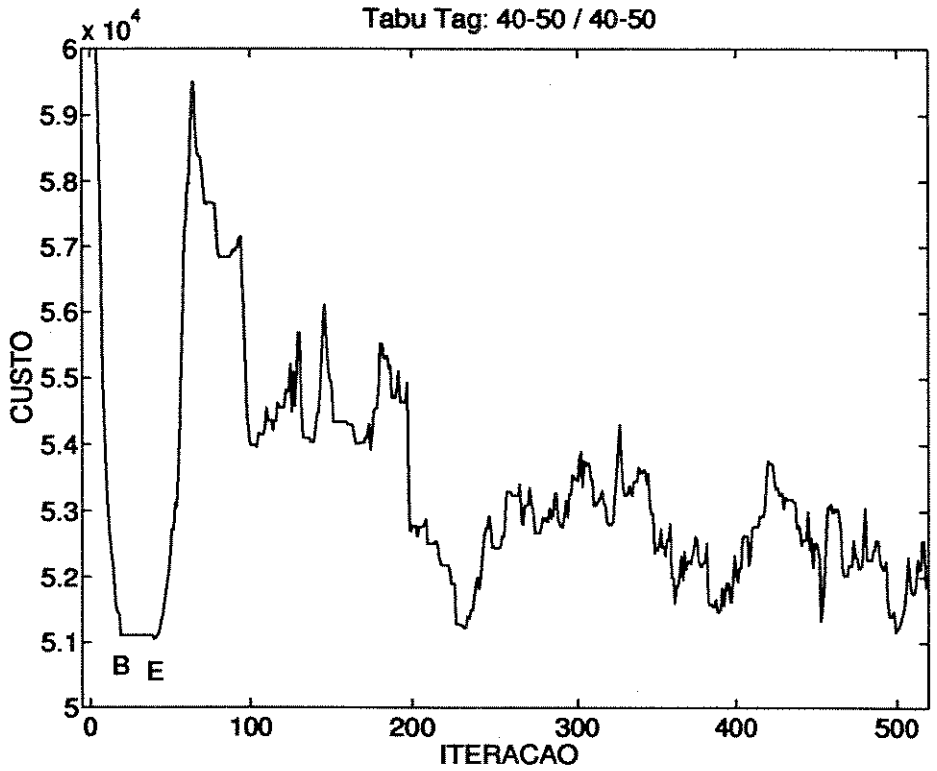

Fig 5.13 - Análise de Intervalos de Geração dos Tabu tags

A figura 5.9 apresenta um processo de busca pouco restritivo, não proibindo por muito tempo arestas caracterizadoras de movimentos recentes, isto faz com que o processo de busca fique preso em uma única região e nesta obtenha sempre soluções com um mesmo valor da função objetivo. Pode-se observar que após o primeiro ótimo local segue-se um pequeno platô. De uma das soluções com o mesmo valor de custo que compõem esse platô consegue-se gerar um vizinho com menor custo, ponto E, e desse só são geradas soluções com o mesmo valor de custo que, devido aos pequenos intervalos de geração dos tabu tags, devem se repetir várias vezes.

Quando se aumenta um pouco o valor dos intervalos de geração dos tabu tags, figura 5.10, já se consegue um pouco de variação. Contudo segue-se uma sequência de platôs com pequena variação entre eles, mostrando que poucas regiões são visitadas, sendo baixa a probabilidade de grandes melhorias. Esses platôs só não apresentam um comportamento cíclico, devido à amplitude dos intervalos de geração dos tabu tags ser razoavelmente grande (25-15=10).

Aumentando-se a grandeza dos intervalos de geração dos tabu tags, aumenta-se

a variação entre as soluções (valor de custo e configurações), como pode ser observado nos intervalos de geração dos tabu tags de 25-35 e 25-40, figuras 5.11 e 5.12 respectivamente. Com essa maior variação são exploradas mais regiões. Essa exploração se dá ativando-se as arestas envolvidas em movimentos recentes por números de iterações, tabu tags, que não sejam pequenos ao ponto de se ficar preso a soluções com um mesmo valor de custo (pode-se observar que existem poucos platôs e os que existem têm pequenas durações). Esses números também não devem ser muito elevados fazendo a busca perder boas características das regiões.

Quando se aumentam muito os intervalos de geração dos tabu tags, a busca torna-se muito restritiva, as arestas caracterizadoras de movimentos recentes ficam mais tempo ativas, podendo proibir movimentos que as utilizem e assim não explorar regiões com boas características. Logo o processo de busca passa a explorar mais espaços de soluções ruins, tanto que para intervalos de geração de tabu tags de 40-50, figura 5.13, a melhor solução é encontrada antes da iteração 50.

É claro que um processo de busca mais restritivo provoca uma maior diversificação e por consequência pode-se acabar explorando regiões mais diversificadas e que possuam melhores soluções. Contudo, esse mecanismo não deve ser utilizado para provocar diversificação, pois ele não tem um procedimento que direcione a busca para regiões que apresentem boas características e quando, por sorte, acaba encontrando uma região de boas soluções, devido ao seu carater restritivo, pode sair rapidamente dessa sem ter encontrado suas melhores soluções.

5.2. Análise de Desempenho da Heurística

Em outro teste foi analisado o comportamento da heurística frente a variação de parâmetros intrínsecos de problemas, como número de famílias, critério de escolha da tarefa para inicialização da construção da solução de partida, faixa de variação das datas de entrega e grandeza dos tempos de preparação de máquina. Para isso foi escolhido um problema com 50 tarefas e para cada combinação de parâmetros foram gerados 10 diferentes instâncias. As tabelas 5.6 a 5.9 apresentam os resultados (média de cada 10 problemas) obtidos para as diferentes combinações. Esses resultados referem-se ao tempo de atraso acumulado, tempo de adiantamento acumulado, aos custos de preparação de máquina, atraso, estoque e ao custo total, todos

referentes a melhor solução encontrada no processo de busca, e às porcentagens de melhoria entre as soluções encontradas em cada parte de heurística. Aqui as porcentagens de melhoria diferem para cada combinação, pois cada uma dessas combinações refere-se a diferentes problemas. Logo tem-se diferentes processos de busca, mesmo na obtenção da solução de partida.

Analisando-se as tabelas 5.6 a 5.9 muitas observações podem ser feitas. Uma delas é que para os pares de casos onde se alteram os critérios de inicialização da solução de partida, pode ser feita uma divisão de comportamento relacionada à dispersão das datas de entrega. Para os casos que apresentam grande dispersão das datas de entrega, com essas sendo geradas entre o instante 10 e 2500 (pares de casos 39-40, 43-44, 47-48 e 51-52), o esforço que a heurística emprega para melhorar a solução de partida, obtida pelo critério de inicialização da ordem com menor data de entrega, é menor do que o empregado quando se utiliza o critério da ordem com maior tempo de processamento, como pode ser observado pelas porcentagens de melhoria entre a solução de partida e o primeiro ótimo local.

Este fato não ocorre quando as datas de entrega estão concentradas mais no final do horizonte de programação, como as geradas entre o instante 500 e 2500 (pares de casos 41-42, 45-46, 49-50, 53-54).

Isto se deve a dois dos componentes dos custos terem como fator gerador a defasagem entre o instante de conclusão e a data de entrega. Quando as datas de entrega estão dispersas pelo horizonte de programação e se inicializa a construção da sequência pela ordem com maior tempo de processamento, essa ordem pode possuir uma data de entrega muito elevada e assim causar um grande custo de estoque, e como esse custo não será mais considerado nas inserções posteriores de ordens para se formar a solução de partida, essa acaba possuindo um grande custo de estocagem, devido a primeira ordem inserida. Já quando as datas de entrega não são tão dispersas e se concentram mais no final do horizonte de programação, independente do critério de inserção da primeira ordem, várias ordens vão ter uma grande defasagem entre seus instantes de conclusão e suas datas de entrega, assim a ordem "mal colocada" pelo critério de maior tempo de procesamento não surtirá um efeito negativo muito grande.

Conclui-se, desta análise, que o critério baseado no maior tempo de processamento não tem nenhum efeito prático e pode inclusive prejudicar o processo de construção da solução de partida. Já o critério de iniciar a construção pela ordem com menor data de entrega, tem o efeito prático de se minimizar as defasagens entre os instantes de conclusão e as datas de entrega. Contudo, se fosse adotado como critério a escolha da ordem com a menor defasagem ter-se-ia

um resultado mais direto e eficiente.

Dados Comuns: tmaxs:80, tmins:10, n:50, h:2500, limit:500, Tabu Tag: ad:25-40 / elim: 25-40

Caso	Composição dos Custos			Tempo Atraso	Tempo Adiantam.	Custo Prepar.	Custo Atraso	Custo Estoque	Custo Total	% de Melhoria		
	α_1	α_2	α_3							Sol-1o.OL	1o.OL-BT	Sol-BT
39	0	0	1	4489	2380	0	0	76563	76563	32.2	4.2	35.0
nf: 5	0	1	0	2046	5845	0	76173	0	76173	83.8	14.0	86.1
tprep: 3	1	0	0	18657	22903	12	0	0	12	0.0	0.0	0.0
datamin: 10	40	0.1	0.1	2255	3840	7753	9199	12724	29676	17.9	2.1	19.6
critsem:	70	0.1	0.1	2020	4452	11745	9080	15191	36016	16.7	2.8	19.0
< data entr.	40	0.5	0.1	1741	5042	6303	38239	18568	63110	17.5	1.3	18.6
	40	0.1	0.5	3371	2521	10537	14119	39686	64342	27.0	2.3	28.6
40	0	0	1	4727	2470	0	0	76310	76310	55.1	2.7	56.3
nf: 5	0	1	0	2727	6718	0	101592	0	101592	87.0	16.8	89.2
tprep: 3	1	0	0	21070	24120	12	0	0	12	0.0	0.0	0.0
datamin: 10	40	0.1	0.1	2247	3593	7550	9838	14722	32110	24.0	5.1	27.9
critsem:	70	0.1	0.1	2217	4552	11199	9874	17894	38967	18.4	4.1	21.7
> t.proces.	40	0.5	0.1	1823	4977	5879	39976	20489	66344	31.0	5.2	34.6
	40	0.1	0.5	3416	2544	10614	14059	39291	63964	44.4	5.8	47.6
41	0	0	1	2021	14859	0	0	416425	416425	20.9	1.7	22.2
nf: 5	0	1	0	360	16419	0	11152	0	11152	96.2	11.4	96.7
tprep: 3	1	0	0	11765	28013	12	0	0	12	0.0	0.0	0.0
datamin: 500	40	0.1	0.1	657	15695	10068	3554	45796	59418	18.9	2.0	20.6
critsem:	70	0.1	0.1	559	16551	15771	3254	49044	68069	15.1	1.0	15.9
< data entr.	40	0.5	0.1	281	16968	7782	7503	53837	69122	16.7	1.2	17.7
	40	0.1	0.5	1451	14609	12546	6994	209059	228599	21.2	1.2	22.1
42	0	0	1	2106	14437	0	0	426249	426249	20.1	2.3	21.9
nf: 5	0	1	0	776	17401	0	21235	0	21235	96.6	1.5	96.7
tprep: 3	1	0	0	13728	28781	12	0	0	12	0.0	0.0	0.0
datamin: 500	40	0.1	0.1	793	15109	10060	3817	46621	60498	18.6	1.8	20.1
critsem:	70	0.1	0.1	532	16054	15561	2972	49186	67719	14.8	3.0	17.4
> t.proces.	40	0.5	0.1	262	16182	7406	6876	55543	69825	18.1	0.7	18.6
	40	0.1	0.5	1433	14353	12259	6719	212353	231331	20.0	3.3	22.7

Tabela 5.6 - Análise da Heurística para Diferentes Problemas

Dados Comuns: tmaxs:80, tmins:10, n:50, h:2500, limit:500, Tabu Tag: ad:25-40 / del: 25-40

Caso	Composição dos Custos			Tempo Atraso	Tempo Adiantam.	Custo Prepar.	Custo Atraso	Custo Estoque	Custo Total	% de Melhoria		
	α_1	α_2	α_3							Sol-1o.OL	1o.OL-BT	Sol-BT
43	0	0	1	19883	142	0	0	5789	5789	9.5	1.1	10.5
nf: 5	0	1	0	6489	5724	0	220825	0	220825	77.3	9.4	79.4
tprep: 10	1	0	0	19132	22672	40	0	0	40	0.0	0.0	0.0
datamin: 10	40	0.1	0.1	4099	3874	12023	18763	12260	43046	17.0	4.1	20.4
critsem:	70	0.1	0.1	4180	6340	18334	19448	18563	56345	14.6	5.4	19.2
< data entr.	40	0.5	0.1	3965	5789	10514	93075	19386	122975	18.6	6.2	23.6
	40	0.1	0.5	7000	591	16792	30164	10721	57677	19.7	2.7	21.9
44	0	0	1	21386	247	0	0	6553	6553	92.7	6.3	93.1
nf: 5	0	1	0	6863	6660	0	238083	0	238083	79.4	16.4	82.8
tprep: 10	1	0	0	21579	23917	40	0	0	40	0.0	0.0	0.0
datamin: 10	40	0.1	0.1	3933	4345	11132	19720	15644	46496	20.2	4.6	23.9
critsem:	70	0.1	0.1	4167	6125	18147	20388	20205	58740	18.9	6.7	24.4
> t.proces.	40	0.5	0.1	3970	5479	9957	91066	20626	121649	28.1	8.0	33.8
	40	0.1	0.5	7141	577	16922	30446	9925	57293	41.3	14.2	49.7
45	0	0	1	14566	5743	0	0	145068	145068	32.4	0.0	32.4
nf: 5	0	1	0	2447	15231	0	64380	0	64380	91.0	9.8	91.9
tprep: 10	1	0	0	12174	27716	40	0	0	40	0.0	0.0	0.0
datamin: 500	40	0.1	0.1	2323	12244	15616	12853	36264	64733	15.2	2.6	17.4
critsem:	70	0.1	0.1	2016	14037	24275	11242	42775	78292	12.9	1.4	14.1
< data entr.	40	0.5	0.1	702	17059	9056	19118	61420	89594	26.5	2.3	28.2
	40	0.1	0.5	6982	6551	23434	33606	78957	135997	31.6	2.1	33.0
46	0	0	1	15966	5405	0	0	146043	146043	43.0	0.2	43.2
nf: 5	0	1	0	3778	16267	0	100758	0	100758	89.2	10.0	90.3
tprep: 10	1	0	0	14174	28515	40	0	0	40	0.0	0.0	0.0
datamin: 500	40	0.1	0.1	2411	11712	15755	13539	37728	67022	13.2	1.5	14.5
critsem:	70	0.1	0.1	1845	13585	23722	10661	44762	79145	13.2	2.8	15.7
> t.proces.	40	0.5	0.1	902	16212	9830	24609	61491	95930	22.0	0.7	22.5
	40	0.1	0.5	7682	5695	24252	35452	76313	136017	30.5	9.8	37.3

Tabela 5.7 - Análise da Heurística para Diferentes Problemas

Dados Comuns: tmaxs:80, tmins:10, n:50, h:2500, limit:500, Tabu Tag: ad:25-40 / elim: 25-40

Caso	Composição dos Custos			Tempo Atraso	Tempo Adiantam.	Custo Prepar.	Custo Atraso	Custo Estoque	Custo Total	% de Melhoria		
	α_1	α_2	α_3							Sol-1o.OL	1o.OL-BT	Sol-BT
47	0	0	1	41347	82	0	0	1985	1985	18.9	0.0	18.9
nf: 30	0	1	0	10813	6224	0	399728	0	399728	77.1	10.3	79.4
tprep: 3	1	0	0	18521	21569	88	0	0	88	0.0	0.0	0.0
datamin: 10	40	0.1	0.1	7301	5097	17057	35003	15632	67692	15.0	2.7	17.4
critsem:	70	0.1	0.1	7438	6949	25848	35518	22335	83701	9.2	5.1	13.9
< data entr.	40	0.5	0.1	6921	6087	15534	168233	20346	204113	27.5	6.9	32.5
	40	0.1	0.5	10813	742	22630	47914	13030	83574	22.5	1.2	23.5
48	0	0	1	44484	106	0	0	2954	2954	96.8	0.0	96.8
nf: 30	0	1	0	11598	8279	0	424310	0	424310	73.7	19.0	78.7
tprep: 3	1	0	0	16295	21170	88	0	0	88	0.0	0.0	0.0
datamin: 10	40	0.1	0.1	7588	5197	16281	36128	18673	71082	20.8	2.7	22.9
critsem:	70	0.1	0.1	7793	7005	25681	36892	22998	85571	16.8	5.5	21.3
> t.proces.	40	0.5	0.1	7901	5866	15783	181006	21998	218787	29.6	2.6	31.4
	40	0.1	0.5	11492	654	22477	49341	15918	87736	35.7	8.9	41.4
49	0	0	1	36266	3024	0	0	75230	75230	28.4	0.0	28.4
nf: 30	0	1	0	4429	14405	0	163407	0	163407	87.9	14.6	89.7
tprep: 3	1	0	0	11757	26808	88	0	0	88	0.0	0.0	0.0
datamin: 500	40	0.1	0.1	4186	13245	19010	21408	40435	80853	11.7	2.4	13.8
critsem:	70	0.1	0.1	3098	17354	26012	18345	54735	99092	7.1	1.4	8.4
< data entr	40	0.5	0.1	2263	15749	14570	58232	57597	130399	34.0	5.6	37.7
	40	0.1	0.5	12166	4050	31422	56195	50871	138488	22.3	3.2	24.8
50	0	0	1	36934	2912	0	0	74522	74522	57.9	0.0	57.9
nf: 30	0	1	0	6975	14860	0	216908	0	216908	82.3	23.3	86.4
tprepar: 3	1	0	0	10065	26943	88	0	0	88	0.0	0.0	0.0
datamin: 500	40	0.1	0.1	3967	12985	19050	21438	43637	84125	15.2	3.5	18.1
critsem:	70	0.1	0.1	3482	16930	25197	18270	59252	102719	9.9	2.2	11.9
> t.proces.	40	0.5	0.1	2734	16436	14760	69574	60501	144835	29.0	7.3	34.2
	40	0.1	0.5	11504	3690	30873	54879	51201	136953	35.3	4.7	38.3

Tabela 5.8 - Análise da Heurística para Diferentes Problemas

Dados Comuns: tmaxs:80, tmins:10, n:50, h:2500, limit:500, Tabu Tag: ad:25-40 / elim: 25-40

Caso	Composição dos Custos			Tempo Atraso	Tempo Adiantam.	Custo Prepar.	Custo Atraso	Custo Estoque	Custo Total	% de Melhoria		
	α_1	α_2	α_3							Sol-1o.OL	1o.OL-BT	Sol-BT
51	0	0	1	149225	45	0	0	1091	1091	43.6	0.0	43.6
nf: 30	0	1	0	24366	9274	0	949380	0	949380	85.3	5.0	86.1
tprep: 10	1	0	0	21855	19774	293	0	0	293	0.0	0.0	0.0
datamin: 10	40	0.1	0.1	18374	9140	26959	84362	34815	146136	14.2	3.0	16.7
critsem:	70	0.1	0.1	19105	13371	38770	88164	56467	183401	11.8	1.4	13.0
< data entr.	40	0.5	0.1	18687	9935	23767	418980	40698	483445	22.8	1.3	23.8
	40	0.1	0.5	24649	1303	43768	114134	23339	181241	22.8	1.6	24.1
52	0	0	1	154542	59	0	0	1485	1485	97.6	25.4	98.2
nf: 30	0	1	0	27255	11356	0	1026473	0	1026473	79.0	8.1	80.7
tprep: 10	1	0	0	19393	19197	293	0	0	293	0.0	0.0	0.0
datamin: 10	40	0.1	0.1	19220	9962	27631	86322	39750	153703	10.2	1.1	11.2
critsem:	70	0.1	0.1	18880	14724	33319	86391	60895	180605	10.8	0.9	11.6
> t.proces.	40	0.5	0.1	19273	10839	27326	432939	45205	505470	23.5	2.4	25.4
	40	0.1	0.5	25489	1781	41817	113704	39285	194806	21.2	7.5	27.1
53	0	0	1	136450	1078	0	0	24623	24623	27.7	0.0	27.7
nf: 30	0	1	0	16478	15211	0	618699	0	618699	89.8	4.2	90.2
tprep: 10	1	0	0	14763	24685	293	0	0	293	0.0	0.0	0.0
datamin: 500	40	0.1	0.1	10196	17226	24567	52828	64165	141560	6.8	1.0	7.7
critsem:	70	0.1	0.1	11378	20013	35749	54879	78916	169544	4.5	0.3	4.8
< data entr.	40	0.5	0.1	10807	16307	24237	255628	66442	346307	28.5	1.9	29.8
	40	0.1	0.5	24409	3869	49548	115762	54169	219479	17.0	1.6	18.3
54	0	0	1	147382	1092	0	0	26303	26303	77.7	5.4	79.0
nf: 30	0	1	0	20599	16330	0	691087	0	691087	85.1	3.1	85.5
tprep: 10	1	0	0	12645	24451	293	0	0	293	0.0	0.0	0.0
datamin: 500	40	0.1	0.1	10056	17161	25353	50748	64816	140917	12.5	1.8	14.0
critsem:	70	0.1	0.1	12706	19739	32953	56593	84694	174240	8.3	1.1	9.4
> t.proces.	40	0.5	0.1	11302	16055	25942	255540	66577	348059	20.0	6.0	24.8
	40	0.1	0.5	24917	3449	50259	118275	61176	229710	25.7	3.6	28.4

Tabela 5.9 - Análise da Heurística para Diferentes Problemas

Pode-se observar, pelas porcentagens de melhoria entre as soluções das 3 etapas da heurística, que esta promove uma grande melhoria até o primeiro ótimo local, indicando que a heurística de contrução da solução inicial não é muito eficiente, gerando soluções ruins que podem ser muito melhoradas. Esta pouca eficiência na construção da solução de partida, está no fato de que se emprega uma adaptação da heurística de Solomon que considera, para cada inserção, apenas uma análise local, 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 um pouco mais custosa e demorada, seria muito melhor constituida.

Observa-se também uma melhoria modesta com o emprego da busca tabu, o que pode indicar que os ótimos locais para PPPUM com tempos de preparação dependentes de sequência não são muito distintos uns dos outros, ou a busca não está conseguindo visitar regiões mais promissoras. Neste caso, um mecanismo de diversificação da busca, que não foi implementado, pode ser de grande importância para se conseguir maiores melhorias.

Aumentando-se a grandeza dos tempos de preparação, aumentam-se não apenas os tempos e custos com preparação de máquina, mas também os tempos e custos totais com atraso, já que maiores tempos de preparação implicam em terminos mais tarde para as ordens, o que aumenta o tempo total de atraso e assim o custo total de atraso, principalmente se as datas de entrega forem pequenas (data de entrega mínima = 10). Isto pode ser observado confrontando-se qualquer composição de custos entre os casos 39 e 43, assim como entre os casos 40 e 44, 41 e 45, 42 e 46, 47 e 51, 48 e 52, 49 e 53 ou entre os casos 50 e 54.

Já os tempos de adiantamento e custos de estoque tendem a diminuir com o aumento dos tempos de preparação de máquina, pela mesma razão que os tempos e custos de atraso aumentam. Isto se acentua quando as datas de entrega são pequenas (data de entrega mínima = 10).

Análise idêntica pode ser feita quando se aumenta o número de famílias, pois esse maior número implica em uma maior incidência dos tempos de preparação, o que também resulta no aumento nos instantes finais de processamento das ordens, aumentando os atrasos e diminuindo os estoques. Isto pode ser observado confrontando-se as composições de custos entre os casos 39 e 47, 40 e 48, 41 e 49, 42 e 50, 43 e 51, 44 e 52, 45 e 53 e entre os casos 46 e 54.

Comparando-se os casos 39 e 41, assim como pela comparação dos casos 40 e 42, 43 e 45, 44 e 46, 47 e 49, 48 e 50, 51 e 53 e finalmente 52 e 54, observa-se que quanto maior

a data de entrega mínima, ou seja, quando as datas de entrega estão mais agrupadas no final do horizonte de programação, menor os atrasos e maiores os adiantamentos e assim menor o custo com atraso e maior o custo com estoque.

Destas observações podem ser feitas as seguintes análises:

1) Se uma empresa apresenta custos de estoque relativamente altos, como na composição α_1=40, α_2=0.1 e α_3=0.5, ela deve planejar bem as vendas evitando a concentração das datas de entrega no final do horizonte de planejamento, ou então trabalhar os clientes para que aceitem o envio dos produtos antes das datas de entrega, podendo até oferecer descontos para tal. Nesta situação o efeito negativo de grandes tempos de preparação de máquina ou de um número elevado de famílias não é tão grande, já que esses tendem a aumentar os instantes de conclusão das ordens, o que diminue o tempo de adiantamento total. Estes aspectos devem ser priorizados com uma produção "folgada", pois neste caso as ordens terminam mais cedo em relação as suas datas de entrega, aumentando ainda mais o custo com estoque.

2) Se a empresa apresenta custo de atraso relativamente alto, como na composição α1=40, α2=0.5 e α3=0.1, ela deve planejar as vendas evitando datas de entrega próximas ao início do horizonte. Deve-se trabalhar também para que os tempos de preparação de máquina sejam pequenos e as ordens estarem agrupadas em poucas famílias, objetivando-se diminuir os instantes de conclusão, diminuindo-se assim o atraso total. Estes aspectos devem ser priorizados para uma situação com uma produção "apertada".

Para cada caso resolvido pela heurística, observa-se que nem sempre há uma correlação entre os comportamentos dos tempos de atraso (adiantamento), com os custos de atraso (estoque), para diferentes composições de custos. Por exemplo, no caso 39 para a composição de custos α_1=0, α_2=1 e α_3=0, tem-se como resultado um tempo total de atraso de 2046 horas e um custo de atraso de 76173 U.M., já para a composição de custos α_1=40, α_2=0.5 e α_3=0.1, tem-se um tempo total de atraso de 1741 horas, menor que para a primeira composição, contudo, tem-se um custo de 38239, proporcionalmente maior que o da primeira composição (α_2 para a primeira composição é o dobro da segunda).

Isto ocorre porque em cada movimento os tempos totais de preparação, atraso e adiantamento podem ser alterados, aumentando ou diminuindo, mas como o objetivo da heurística não é a minimização de tempos mas sim de custos, pode ocorrer de uma solução que possua um tempo de atraso (adiantamento) menor que uma outra, tenha um custo de atraso (estoque) maior que essa segunda, pois está se considerando que além dos tempos também as quantidades

compõem tais custos.

Em um movimento pode ocorrer, portanto, de um tempo de atraso que tenha aumentado estar associado a uma grande quantidade e mesmo que a soma dos tempos de atraso tenha diminuido, associado aos tempos de atraso que diminuiram, estão pequenas quantidades, resultando que o custo total tenha aumentado.

Pode ocorrer ainda em cada movimento as seguintes variações:

- Tempo total de atraso (adiantamento) diminuiu e custo de atraso (estoque) diminuiu;
- Tempo total de atraso (adiantamento) aumentou e custo de atraso (estoque) aumentou;
- Tempo total de atraso (adiantamento) aumentou e custo de atraso (estoque) diminuiu;

A solução da heurística será um programa de produção onde os valores de tempo de atraso (adiantamento) e custo de atraso (estoque) são resultados de vários movimentos executados. Logo, conforme os dados e parâmetros de cada problema, tem-se diferentes sequências de movimentos e para cada sequência de movimento tem-se uma solução, onde o custo de atraso (estoque) considera, além dos tempos de atraso (adiantamento), as quantidades de produtos que serão entreques com atraso (que ficarão estocadas).

No caso 40, para as mesmas composições de custos, observa-se que o tempo de atraso da segunda composição é menor que o da primeira e que o mesmo ocorre, proporcionalmente, com os custos de atraso. Mesmo se considerando a diferença entre sequência de movimentos na obtenção de cada solução, não parece lógico que uma solução obtida se considerando apenas o custo de atraso, possua um custo de atraso total maior que uma solução obtida considerando-se também os componentes de custo de preparação e estoque. É que além da sequência de movimentos também pode estar contribuindo para tal fato a obtenção da solução de partida. A fase de construção da solução de partida verifica o custo apenas da ordem a ser inserida, portanto ao se inserir uma ordem na sequência esta pode estar implicando em um grande atraso das outras ordens subsequentes à inserção. E mesmo com a diversificação que a heurística proporciona, o número de iterações pode não ser o suficiente para sair de um sub-espaço de soluções ruins.

Este fato revela que mesmo em algoritmos dotados de mecanismos de busca tabu, a solução de partida pode ter um papel relevante na qualidade das soluções geradas. Isto torna a alteração da fase de construção da solução de partida, já sugerida, de grande importância para uma maior eficiência da heurística.

Para todos os casos quando se tem composições de custos ponderando apenas o

custo de estoque ou o custo de atraso(α_1=0, α_2=0, α_3=1 e α_1=0, α_2=1, α_3=0, respectivamente), a heurística emprega um grande esforço para a obtenção da solução, em especial até a obtenção do primeiro ótimo local. Por que isto ocorre?

Em cada uma dessas composições não se tem objetivos antagônicos. Portanto, quando se pondera apenas o custo de estoque, ao se executar um movimento que aumente ou crie tempos de preparação, diminui-se muitos os tempos de adiantamento seguintes a tal aumento, outros se transformam em tempos de atraso e muitos tempos de atraso são aumentados. 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, só que a grande diminuição do mesmo se dará quando se diminuir ou eliminar tempos de preparação de máquina, já que não há a ponderação do custo antagônico de estoque.

Sendo assim, a heurística também é aplicável, e com boa eficiência, a problemas cuja medida de desempenho objetive apenas minimizar custos de atraso ou custos de estoque.

Para a composição de custos que pondera apenas o custo de preparação de máquina (α_1=1, α_2=0, α_3=0), sempre ocorre de não haver melhoria após a solução de partida.

Isto se deve aos tempos de preparação entre famílias serem gerados de forma proporcional à diferença entre essas famílias. Na fase de construção da solução de partida vai-se inserindo as ordens a fim de se minimizar custos locais, no caso custos de preparação, iniciando-se a programação com uma ordem pertencente a uma família **f**, que obedeça um certo critério. Todas as ordens pertencentes a essa mesma família se inseridas juntas a primeira ordem implicam em custo zero. Logo, insere-se todas as ordens da família **f**.

Após isto insere-se uma ordem que produza o menor custo, o que ocorre quando essa nova ordem for inserida ou entre a ordem zero e a primeira ordem programada, ou entre a última ordem programada e a ordem zero. Como o objetivo é apenas minimizar os custos de preparação, essa nova ordem a ser inserida pertencerá a uma família **f'**, adjacente à família **f**. E como no início, a ordem que será inserida em seguida a essa também pertencerá à família **f'**. E assim por diante até se programar todas as ordens dessa nova família.

Esse processo continua até que todas as ordens tenham sido inseridas. Como resultado, tem-se uma sequência de famílias, começando da primeira ou da última e seguindo até todas as ordens estarem programadas. A diferença entre cada família é de 1 e, assim, o custo total é de 1*tprep*fator de aleatoriedade (± 0.05) * (número de famílias -1).

5.3. Número de Famílias

Devido a necessidade cada vez maior de diversificação da produção, com um maior número de tipos de produtos a serem oferecidos ao mercado, as empresas têm empregado esforços para uma maior flexibilidade da manufatura. Esses esforços começam pela adoção e implantação de formas de organização tais como: células flexíveis de manufatura e tecnologia de grupo, objetivando-se a redução de tempos de processamento e de preparação de máquina.

Contudo, muitas empresas ainda não fizeram a implementação de tais metodologias e mesmo algumas que a fizeram ainda se deparam com um número relativamente alto de famílias de peças que possuem diferentes preparações de máquina.

Surge então a questão de como irá se comportar a heurítica frente a problemas com um número relativamente pequeno de famílias e frente a problemas em que esse número seja bem maior.

Através da calibragem de parâmetros, a heurística deve-se adequar a diferentes estruturas de problemas, já que diferentes estruturas implicam em diferentes espaços de solução e esses requerem processos de busca que melhor se adaptam a suas características. Para um problema com um pequeno número de famílias, há uma maior incidência de soluções com um mesmo valor de custo, necessitando o processo de busca ser mais restritivo, a fim de sair dos platôs.

Essa análise pode ser confirmada através de dois testes simples em dois problemas com 50 ordens de produção. No problema 1 as ordens estão agrupadas em 20 famílias e o multiplicador tprep tem valor 5. No problema 2 as ordens estão agrupadas em 5 famílias e o multiplicador tprep tem valor 20.

O primeiro teste refere-se ao parâmetro "intervalos de variação dos tabu tags". Os intervalos e resultados são apresentados na tabela 5.10. O segundo teste avaliou o comportamento da heurística frente a tais problemas para diferentes maxted, tabela 5.11.

Problema	Tabu Tag		Custo Prepar.	Custo Atraso	Custo Estoque	Custo Total	% de Melhoria	
	ad	elim					1ºOL - BT	Sol - BT
1	10 - 20	10 - 20	1318	2610	2876	6804	2.3	15.9
	20 - 30	20 - 30	1294	2569	2784	6647	4.6	17.9
	25 - 35	25 - 35	1315	2621	2715	6651	4.5	17.8
	35 - 40	35 - 40	1349	2601	2669	6619	5.0	18.2
2	10 - 20	10 - 20	1270	2493	2795	6558	1.4	13.2
	20 - 30	20 - 30	1275	2511	2763	6549	1.5	13.4
	25 - 35	25 - 35	1270	2477	2828	6575	1.2	13.0
	35 - 40	35 - 40	1258	2449	2746	6453	3.0	14.6

Tabela 5.10 - Número de Famílias e Tabu Tag

Problema	maxted	Custo Prepar.	Custo Atraso	Custo Estoque	Custo Total	% de Melhoria	
						1ºOL - BT	Sol - BT
1	1	1305	3877	2527	7709	2.83	17.61
	2	1150	3552	2569	7271	8.35	22.29
2	1	1091	3408	2024	6523	14.60	21.08
	2	1152	4338	2148	7638	0.00	7.59

Tabela 5.11 - Número de Famílias e maxted

Pela observação das tabelas 5.10 e 5.11 confirma-se a necessidade de se restringir mais o processo de busca quando o número de famílias é menor. Na tabela 5.10 vê-se que o primeiro problema consegue maior melhoria e que ao se passar de tabu tag 10-20 para 20-30 se dá um salto em termos de qualidade de solução. Já para o segundo problema, além de não se conseguir uma melhoria tão expressiva, o salto de qualidade só ocorre com um tabu tag bem maior, 35-45. Na tabela 5.11 o problema com um número maior de famílias é melhor resolvido empregando-se maxted = 2, menos restritivo que maxted =1. O inverso ocorre com o problema com um número menor de famílias. Estes dois testes mostram realmente que para um número de famílias pequeno há a necessidade de se empregar um processo de busca mais restritivo, forçando a saída de regiões com soluções pouco distintas.

O fato de se conseguir uma porcentagem de melhoria maior em problemas com um número maior de famílias, confirma a vantagem em se agrupar o máximo possível as ordens em poucas famílias, pois essa maior porcentagem ocorre devido a maior possibilidade de obtenção de soluções com diferentes combinações entre ordens de diferentes famílias, implicando assim na maior possibilidade de se melhorar uma dada solução, ou seja, problemas com muitas famílias possuem um número maior de soluções ruins do que problemas com poucas famílias.

5.4. Mudança de Atributo

O comportamento do processo de busca é ditado pelo mecanismo de pertubação adotado e pelo atributo que caracteriza os movimentos proibidos.

Como mencionado, os atributos arestas eliminadas e arestas adicionadas definem movimentos realizados com o 2-opt e, por esta razão, eles foram empregados para caracterizar movimentos proibidos. Contudo, devido a características dos problemas estudados, esses atributos não impedem a ocorrência de diferentes soluções com um mesmo valor da função de custo. Essas ocorrências são minimizadas com o uso de parâmetros adequados ao problema.

Um outro atributo que pode ser usado e que não cai em tal problema é o valor da função de custos, pois proibindo-se movimentos que resultem em um mesmo valor de custo, o processo de busca deve se direcionar para uma outra região de custo, distinta da recém explorada.

Fez-se então tal modificação na heurística e empregou-se essa nova versão para a resolução de um problema com 50 ordens, adotando-se diferentes intervalos de geração de tabu

tag, responsáveis agora pela proibição de valores de custo.

As figuras 5.13 a 5.16 apresentam o comportamento do processo de busca para tais resoluções.

Esse mesmo problema foi resolvido pela heurística originalmente proposta e as soluções encontradas podem ser observadas na figura 5.17.

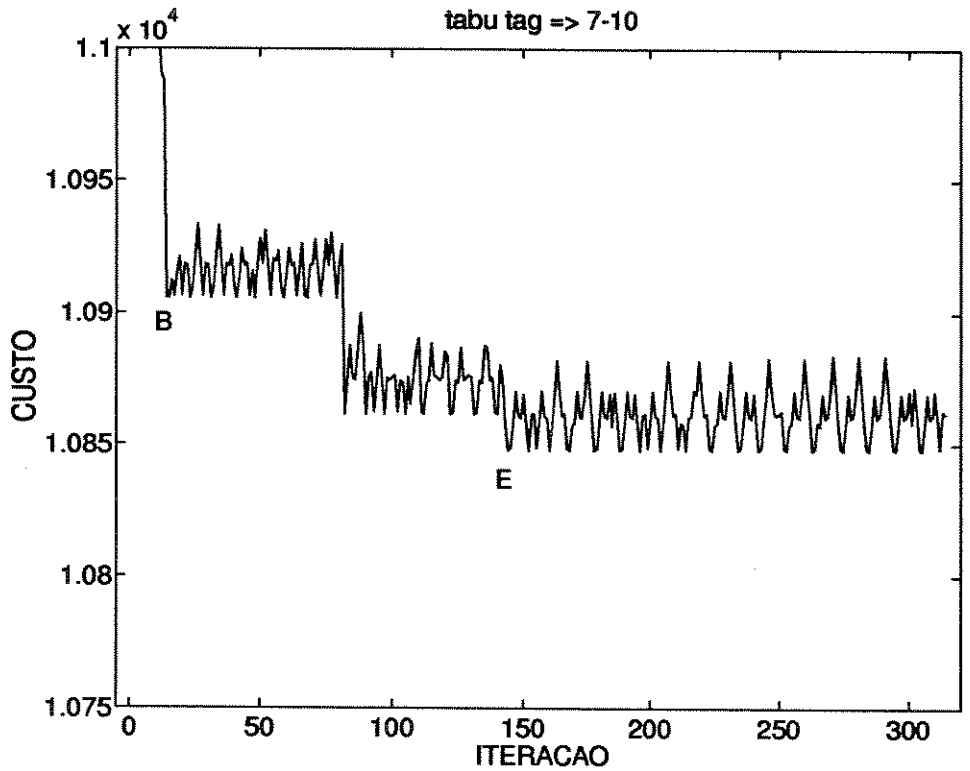

Fig. 5.13 - Atributo Custo

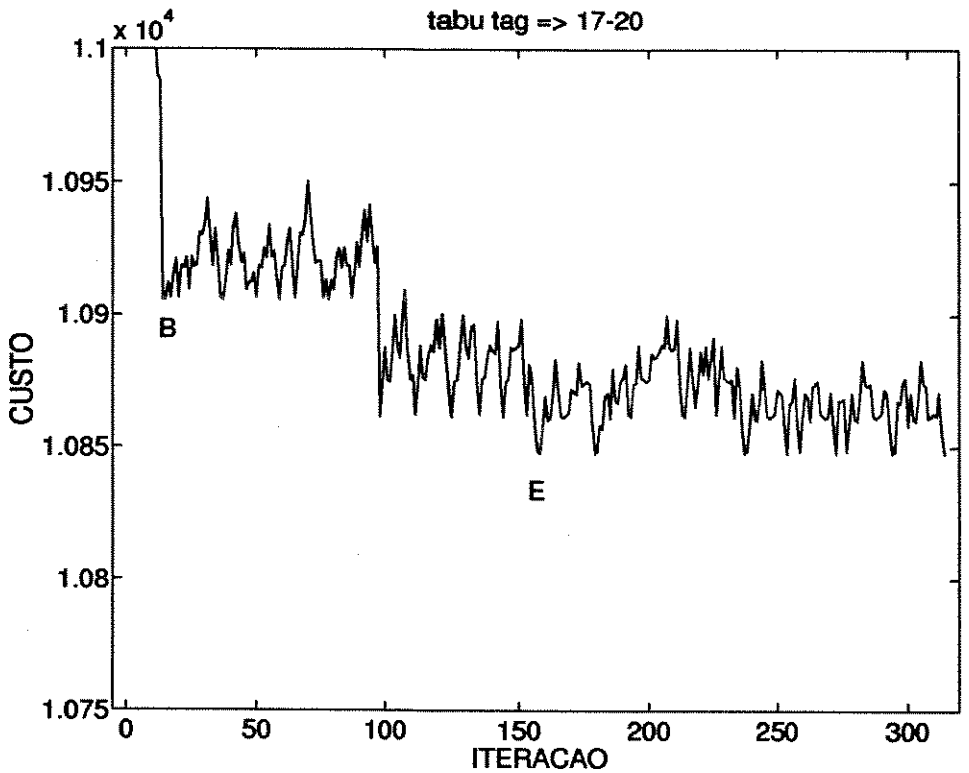

Fig. 5.14 - Atributo Custo

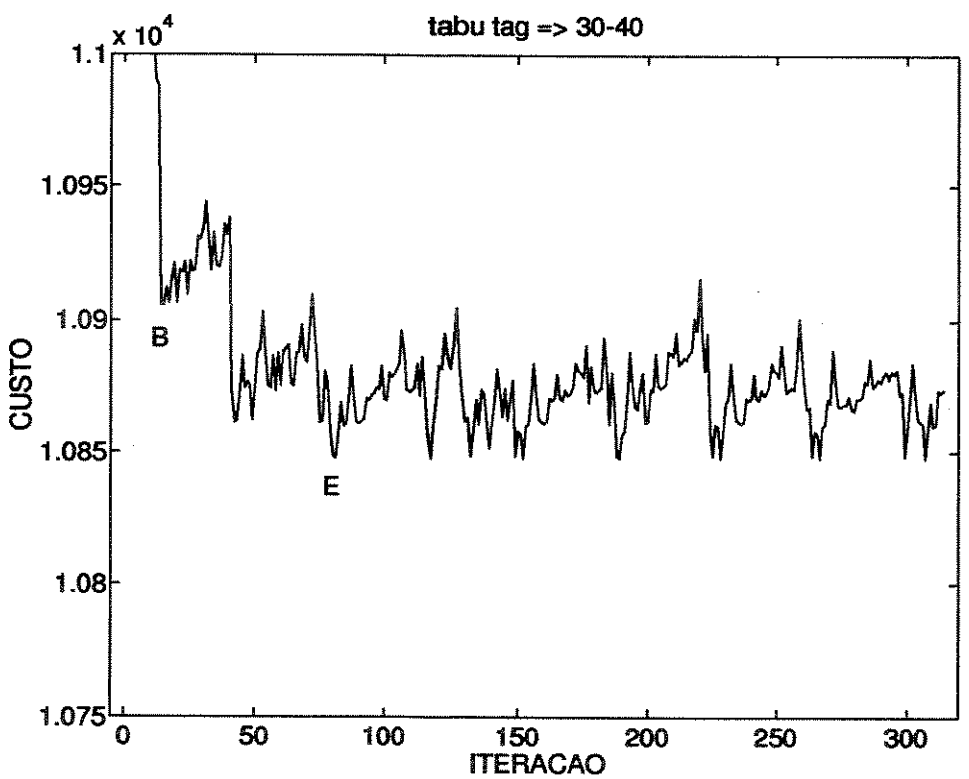

Fig. 5.15 - Atributo Custo

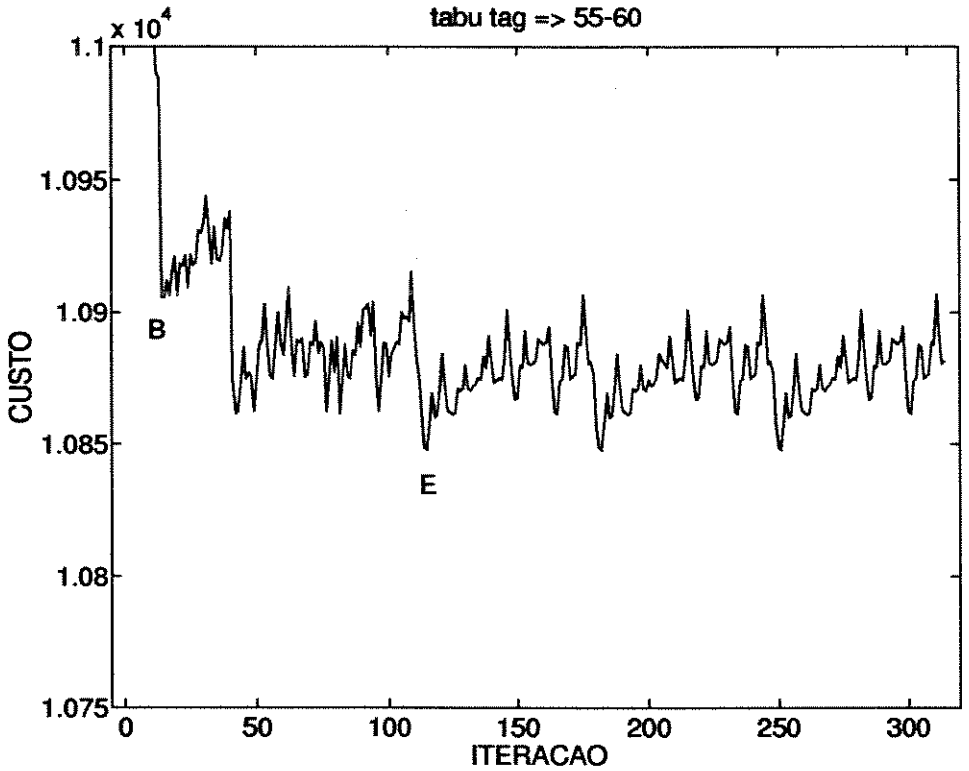

Fig. 5.16 - Atributo Custo

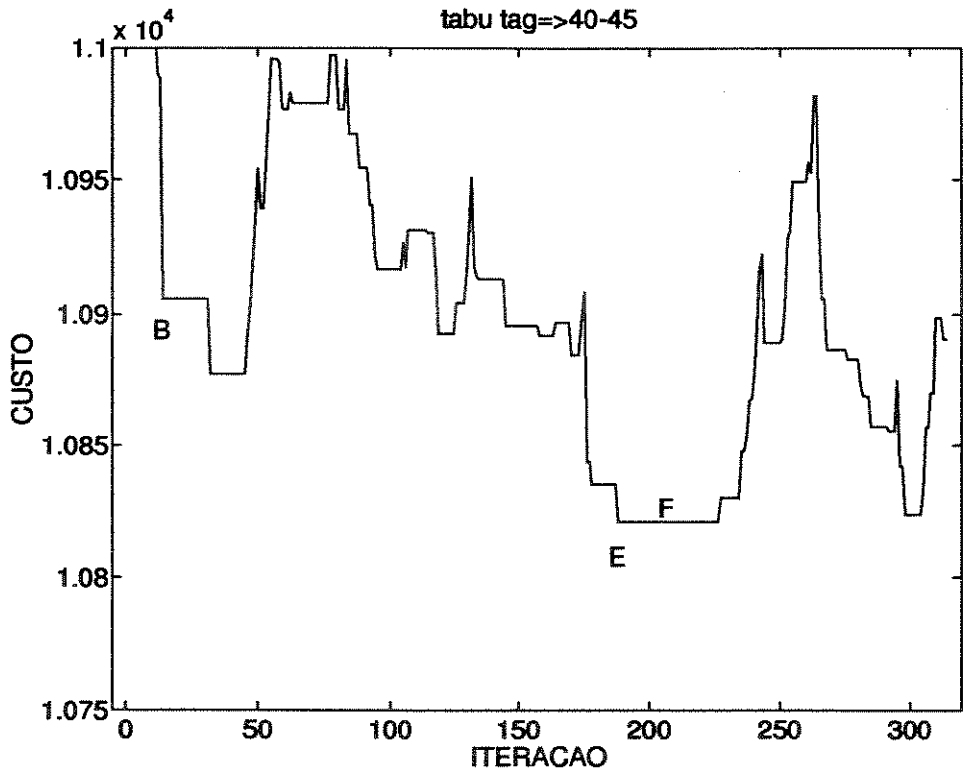

Fig. 5.17 - Atributo Aresta

Analisando os gráficos referentes ao processo de busca que emprega o atributo custo, observa-se que a variação entre as soluções encontradas aumenta conforme aumenta o intervalo de geração dos tabu tags (busca mais restritiva). Contudo, essa maior variação não é suficiente para a obtenção de uma melhor solução, já que em todas as resoluções observa-se a solução com um mesmo custo, 10847.75. Ao se confrontar o comportamento dessas buscas, que empregam o atributo custo, com a busca que emprega o atributo arestas, ve-se que a proibição de soluções com um custo obtido em uma solução recente, prende a busca em soluções bem próximas umas das outras e quando se utiliza o atributo arestas há uma maior variação entre as soluções obtidas.

Quando se sai de um platô, devido às arestas proibidas, caracterizadoras de soluções próximas, "dá-se um salto" para regiões que possuem características distintas, o que não ocorre quando se emprega o atributo custo, que não permite diversificação da busca.

O emprego do atributo custo revela-se, portanto, inferior ao atributo aresta. Para evitar soluções com um mesmo custo pode se aplicar o atributo aresta após uma análise prévia do custo das soluções geradas. Isto é, se a solução gerada tiver custo igual à solução que a gerou descarta-se tal solução. Das soluções não descartadas analisam-se as que não são tabu pelo atributo aresta e escolhe-se a melhor.

Se se optar pelo uso do atributo custo, o uso de mecanismos de diversificação faz-se muito importante.

5.5. 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 aplicadas em PPP 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 sequência crescente de tempos de processamento. Ela resulta em boas soluções quando empregada em problemas cuja medida de desempenho seja: minimização do número médio de jobs no sistema,

ou minimização do tempo de permanência médio das ordens no sistema, ou minimização do trabalho em processo.

A regra EDD consiste na programação das ordens em uma sequência crescente das datas de entrega. Ela apresenta bons resultados quando se objetiva minimizar o desvio entre a data de entrega e o instante de conclusão das ordens.

Contudo, bons resultados na aplicação dessas regras podem ocorrer em problemas onde os tempos de preparação de máquina não são dependentes da sequência e inclusos nos tempos de processamento.

Para se verificar a eficiência da heurística desenvolvida, frente a tais regras de despacho, resolveu-se um mesmo PPPUM com tempos de preparação dependentes da sequência. Os resultados obtidos pela heurística, assim como pelas regras de despacho, estão apresentados na tabela 5.12. Nesses resultados, os custos apresentados foram todos calculados da mesma forma, através das funções já apresentadas.

	Custo				Tempo	
	Prepar.	Atraso	Estoque	Total	Atraso	Estoque
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

Tabela 5.12 - Regras de Despacho X Heurística

As regras de despacho EDD e SPT não devem ser utilizadas na resolução de PPPUM com tempos de preparação depentes da sequência, obviamente, Devido ao fato da heurística ter sido desenvolvida considerando-se os tempos de preparação dependentes da sequência, essa apresenta uma eficiência muitíssimo superior às regras de despacho.

A regra EDD apresenta um desempenho um pouco superior à SPT, pois visa a redução das defasagens entre datas de entrega e instantes de conclusão, formadoras dos custos de atraso e estoque. Contudo, esse sequenciamento simples pode levar à existência de muitos tempos de preparação entre ordens, o que acaba elevando os instantes de conclusão das ordens

, aumentando em muito os atrasos, o que pode ser facilmente visto na tabela 5.12.

Sequenciar as ordens pelos tempos de processamento, não tem grande aplicação quando se objetiva a redução de custos de preparação de máquina, atraso e estoque, a não ser quando se tem uma certa ociosidade na produção e custos de atraso altos, ou quando a produção está apertada e os custos de estoque são altos, pois como já foi visto, com a ociosidade aumentam-se os tempos de adiantamento e diminuem-se os de atraso. Quando a produção está apertada ocorre o contrário.

5.6. Análise da Heurística em uma Situação Prática Real.

Até o momento analisou-se a heurística em problemas cujos dados (tempos de processamento, tempos de preparação, datas de entrega e agrupamento em famílias) foram gerados de forma aleatória. Na prática pode se ter tais dados variando de formas diferentes.

Para a análise da heurística em uma situação real de PPPUM com tempos de preparação dependentes da sequência, obteve-se, por intermédio do Prof. Dr. Paulo C. Lima da Faculdade de Engenharia Mecânica da UNICAMP, dados reais referentes a uma empresa de chapas poliméricas. Antes da resolução desse problema pela heurística, devem ser feitos alguns comentários a respeito do sistema de manufatura a ser programado.

As chapas a serem produzidas podem diferir uma das outras basicamente pela cor, espessura, comprimento e largura. O processo produtivo pode ser resumido da seguinte forma: a matéria prima, geralmente em estado granulado, é introduzida na máquina extrusora através de pequenos *bins* de admissão. Essa máquina extrusora possue uma câmara de formato cilíndrico, a qual é aquecida. A câmara contém uma rosca sem fim que irá empurrar o polímero para a outra extremidade da câmara. Durante o trajeto pela câmara, o polímero se funde e o fato de estar fundido possibilita que este tome o formato do molde (chapa). Essa chapa segue por uma esteira e nesta será cortada por uma guilhotina. Conforme a largura e o comprimento da chapa ela pode ter três diferentes arrumações:

- O molde define a largura da chapa e a guilhotina o comprimento.
- A guilotina define a largura da chapa e o molde comprimento.
- Para pequenas larguras pode-se produzir duas chapas simultaneamente, acoplando uma faca ao molde. Assim o molde define as larguras das duas capas e a guilhotina o

comprimento.

Essas diferenças na arrumação podem resultar em tempos menores de preparação, e/ou processamento.

Calcula-se previamente os tempos de processamento para cada arrumação possível de cada ordem (quantidade de chapas com as mesma característica encomendadas por um mesmo cliente e com uma mesma data de entrega), em função das características das chapas e quantidades a serem produzidas. Os tempos de preparação entre cada arrumação possível das ordens, também é calculado em função das características dessas ordens.

Esse problema prático diferencia-se dos problemas teóricos analisados até então devido a diferentes arrumações possíveis para cada ordem. Portanto a heurística deve ser adaptada para ser capaz de analisar tal situação em cada uma das suas fases.

Para a fase de construção da solução de partida, analisam-se em cada inserção todas as arrumações possíveis para as ordens ainda não programadas e a arrumação que apresentou menor custo local será inserida na posição correspondente a tal custo.

Os tempos de preparação entre ordens são causados por diferenças entre cor, largura ou espessura, de duas ordens adjacentes no programa. Os tempos causados por diferenças de comprimento são desprezíveis. A matriz de tempos de preparação não é simétrica, já que o tempo de preparação entre uma ordem de cor clara e uma ordem de cor escuta, é bem maior que entre a mesma ordem de cor escura seguida pela ordem de cor clara.

Nas fases de melhoria e de busca tabu, com a realização de um movimento 2-opt, algumas ordens mudam de lugar na sequência, causando novos tempos de preparação de máquina. A ordem posterior à primeira aresta eliminada troca de lugar com a aresta anterior à segunda aresta eliminada e a sequência das ordens entre essas é invertida (Figura 3.4). Devido à troca de lugar dessas duas ordens e à não simetria da matriz de tempos de preparação, novas arrumações das ordens a partir da primeira aresta eliminada podem resultar em um custo total menor que o emprego das ordens antigas. Contudo, se a cada movimento for necessário analisar todas as mudanças de arrumações possíveis para todas as ordens posteriores à primeira aresta eliminada, o tempo de execução da heurística seria inviável. Portanto, o que se fez foi encontrar o melhor movimento não tabu e, a seguir, verificar quais as melhores arrumaçòes para as ordens que causaram maior mudança com o movimento (a ordem posterior a primeira aresta eliminada e a ordem anterior a segunda aresta eliminada).

Após essas alterações 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 na resolução desse problema encontram-se na tabela 5.13 a seguir. Nas resoluções não foram utilizados os custos reais de preparação de máquina, atraso e estoque, já que não se teve acessos aos mesmos. Utilizou-se portanto duas composições de custo para cada caso para se verificar a eficiência da heurística em diferentes cenários.

Máquina	Nº de Ordens	Custo Prepar.	Custo Atraso	Custo Estoque	Custo Total	% de Melhoria		
						Sol-1OL	1OL-BT	Sol-BT
$\alpha_1 = 4, \alpha_2 = 0.01, \alpha_3 = 0.01$								
1	42	1920	119428	14517	135865	59.1	25.2	69.4
3	115	5960	906301	28154	940415	63.8	1.4	64.3
$\alpha = 2, \alpha_2 = 0.01, \alpha_3 = 1$								
1	42	1130	159122	20768	181020	34.6	17.9	46.3
3	115	3890	1212271	832	1216993	51.4	2.1	52.4

Tabela 5.13 - Heurística em um Problema Real de Programação

Verifica-se que o comportamento assumido 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 dos parâmetros, o que pode ter resultado no fraco desempenho da fase tabu, contudo, percebe-se que o método 2-opt obtém bons resultados para esse problema, já que ocorre uma grande melhoria até a obtenção do primeiro ótimo local.

5.7. Conclusões Gerais

Os estudos e os testes computacionais desenvolvidos durante esta tese permitiram uma série de conclusões. As de maior destaque são apresentadas a seguir:

- A principal contribuição deste trabalho está no estudo e na proposta de resolução de

problemas de programação da produção em uma máquina considerando-se:

· Agrupamento de ordens em famílias de ajuste de máquina;

· Tempos de preparação dependentes da sequência;

· Ponderação dos custos de atraso, estoque e preparação de máquina como medida de desempenho dos programas de produção gerados;

· A possibilidade de se interferir na elaboração dos custos e nas datas de entrega, possibilitando a obtenção de diferentes resultados para análise e negociação com clientes.

- Dos resultados obtidos tem-se várias conclusões quanto ao uso da heurística e dos métodos empregados nas suas fases. As principais são:

· Quanto menor o problema mais relaxada deve ser a busca, pois do contrário não sobram muitos vizinhos para se prosseguir com a busca e os vizinhos que sobram têm uma menor possibilidade de possuirem caracteríticas de boas soluções;

· Deve-se encontrar um bom "mix" de valores de parâmetros usados na busca. Para valores de maxted muito baixos e intervalos de geração de "tabu tags" muito altos a busca é muito restritiva, deixando de explorar regiões com boas características. Para valores de maxted muito altos e intervalos de geração de "tabu tags" muito baixos ocorre o contrário; aceitam-se muitas soluções com as mesmas características de soluções recentes, fazendo com que haja pouca variação no espaço de soluções, diminuindo muito a probabilidade de se encontrar uma solução muito boa em outra região do espaço de soluções. A definição de limit deve ser feita considerando-se que a probabilidade de se encontrar melhores soluções irá diminuindo com o processo de busca e assim uma busca muito demorada pode não resultar em uma solução final muito melhor que uma busca de menor duração.

· Tem-se, da conclusão anterior, que um processo de busca mais restritivo provoca uma maior diversificação. Contudo, esse mecanismo de diversificação não tem um procedimento que direcione a busca para regiões que apresentem boas soluções, encontrando poucas vezes tais regiões. Logo deve-se utilizar os mecanismos citados em 3.2.2 para se provocar diversificação.

· 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. Neste caso, deve-se evitar datas de entrega próximas ao início do horizonte de programação.

· Se os tempos de preparação de máquina ou o número de famílias forem pequenos, os instantes de término das ordens são menores, o que resulta em tempos e custos de atraso menores e tempos de adiantamento e custos com estoque maiores. Neste caso, deve-se evitar ordens com datas de entrega concentradas no final do horizonte de planejamento.

· A heurística também é aplicável a problemas cuja medida de desempenho objetive apenas minimizar custos de atraso ou custos de estoque. E com boa eficiência.

· Quanto menor o número de famílias, mais restritivo deve ser o processo de busca, forçando a saída de regiões com soluções pouco distintas.

· A utilização do atributo custo impede a ocorrência de platôs, contudo, prende a busca em soluções bem próximas umas das outras, resultando em um processo de busca com poucas variações e que dificilmente explorará diferentes regiões. O ideal para se evitar platôs e se ter uma maior diversificação é se ter uma união ou "mistura" dos dois atributos.

· As regras de despacho EDD e SPT mostram-se muito pobres na resolução de PPPUM com tempos de preparação dependentes da sequência.

· 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.

5.8. Perspectivas

Mesmo tendo a heurística apresentando bons resultados e estar considerando aspectos importantes e poucas vezes considerados na resolução de problemas de programação da produção, foram levantados ao longo do trabalho e das conclusões outros aspectos que podem ser considerados para pesquisas futuras. Esses aspectos podem ser interessantes para uma maior eficiência da heurística, ou para considerações que a heurística ainda não leva em conta. Os

aspectos mais interessantes para serem considerados em projetos futuros são:

- Consideração de classes distintas para os clientes, onde alguns clientes podem ter uma prioridade maior que outros.
- Implementação de mecanismos de diversificação e intensificação na fase de busca e critério de aspiração.
- Alteração do algoritmo de construção da solução inicial, para que esse passe a calcular o custo total em cada inserção e não apenas o local. O critério de inicialização da solução de partida também deve ser alterado para que a primeira ordem a ser inserida seja a que possue a menor defasagem entre o instante de conclusão e a data de entrega.
- Poder fixar as datas de entrega de algumas ordens, onde a violação dessas não resulte em penalidade por atraso mas em infactibilidade do programa de produção. Isto para maior poder de decisão e negociação com os clientes.
- Consideração de tempos de ociosidade como variáveis de decisão, já que a inserção desses em um programa de produção provoca uma composição de custos de atraso e estoque diferente.
- Resolução do PPPUM com tempos de preparação dependentes da sequência via programação multi-objetivo, já que a função objetivo proposta é uma combinação de custos antagônicos.
- Aplicar o atributo aresta após o emprego do atributo custo, evitando-se a ocorrência de platôs e aumentando-se ainda mais a diversificação do processo de busca.

Referências Bibliográficas

Bagchi, U., Chang, Y., Sullivan, R.S., (1987), "Minimizing Absolute and Squared Deviations of Completion Times with Different Earliness and Tardiness Penalties and a Common Due date", *Naval Research Logistics*, Vol. 34, Nº 5, pp. 739-751.

Bai, S.X. e Gershwin, S.B., (1990), " Scheduling Manufacturing Systems with Work-in-Process Inventory", *Proceedings of the 29th Conference on Decision and Control*, IEEE, pp.557-567.

Baker, K.R., (1974), *Introduction to Sequencing and Scheduling*, John Wiley & Sons.

Baker, K.R. e Scudder, G.D., (1990), "Sequencing with Earliness and Tardiness Penalties: A Review", *Operations Research*, Vol. 38, Nº 1, pp. 22-36.

Berretta, R.E., (1993), "Otimização do Planejamento da Produção em Sistemas Multiestágios", Tese de Mestrado, Faculdade de Engenharia Elétrica, UNICAMP.

Bianco, L. Ricciardelli, S., Rinaldi, G. e Sassano, A., (1988), "Scheduling Tasks with Sequence-Dependent Processing Times", *Naval Research Logistics*, Vol. 35, pp. 177-184.

Blazewicz, J., Dror, M. e Weglarz, J., (1991), "Mathematical Programming Formulations for Machine Scheduling: A Survey", *European Journal of Operational Research*, Vol. 51, pp. 283-300.

Blazewicz, J., Ecker, K., Schmidt, G. e Weglarz, J., (1993), *Scheduling in Computer and Manufacturing Systems*, Springer.

Bodin, L., Golden, B.L., Assad, A.A. e Ball, M., (1983), "The State of the Art in the Routing and Scheduling of Vehicles and Crews", *Computers and Operations Research*, Vol. 9, pp.63-212.

Cerny, V., (1985), "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", *Journal of Optimization Theory and Applications*, Vol. 45, pp. 41-51.

Cheng, T.C.E., (1991), "Optimal Assignment of Slack Due-Date and Sequencing of Jobs with Random Processing Times on a Single Machine", *European Journal of Operational Research*, Vol. 51, pp. 348-353.

Cheng, T.C.E. e Gupta, M.C., (1989), "Survey of Scheduling Research Involving Due Date Determination Decisions", *European Journal of Operational Research*, Vol. 38, pp. 156-166.

Cheng, T.C.E. e Kahlbacher, H.G., (1991), "A Proof for the Longest-Job-First Policy in One-Machine Scheduling", *Naval Research Logistics*, Vol. 38, pp. 715-720.

Desrochers, M., Lenstra, J.K., Savelsbergh, M.W.P., Soumis, F., (1988), "Vehicle Routing with Time Windows: Optimization and Approximation", em *Vehicle Routing: Methods and Studies*, Golden, B.L. and Assad, A.A. (editors), Elsevier Science Publishers B.V..

Eglese, R.W., (1990), "Simulated Annealing: A Tool for Operational Research", *European Journal of Operational Research*, Vol. 46, pp. 271-281.

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

Fry, T.D., Leong, G.K. e Rakes, T.R., (1987), "Single Machine Scheduling: A Comparison of Two Solution Procedures", *OMEGA*, Vol. 15, Nº 4, pp. 277-282.

Gendreau, M., Laporte, G. e Solomon, M.M., (1991), "Sequence Dependent Scheduling to Minimize the Number of Late Jobs", *Centre de Recherche Sur Les Transports* - Publication # 771, Université de Montréal.

Glover, F., (1977), "Heuristics for Integer Programming Using Surrogate Constraints", *Decisions Science*, Vol. 8, pp. 156-166.

Glover, F., (1989), "Tabu Search, Part I", *ORSA Journal on Computing*, Vol. 1, Nº 3, pp.190-206.

Glover, F., (1990), "Tabu Search, Part II", *ORSA Journal on Computing*, Vol. 2, Nº 1, pp. 4-32.

Glover, F., Laguna, M., Taillard, E. e de Werra, D., (1993), "Tabu Search", *Annals of Operations Research*, Vol.41.

Hoogeveen, J.A. e van de Velde, S.L., (1991), "Scheduling Around a Small Common Due Date", *European Journal of Operational Research*, Vol. 55, pp. 237-242.

Kirkpatrick, S., Gellat Jr., C.D. e Vecchi, M.P., (1983), "Optimization by Simulated Annealing", *Science*, Vol. 220, pp. 671-680.

Laporte, G., (1992), "The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms", *European Journal of Operational Research*, Vol. 59, pp. 231-247.

Lawrence, S.R., (1991), "Scheduling a Single Machine to Maximize Net Present Value", *International Journal of Production Research*, Vol. 29, Nº 6, pp. 1141-1160.

Lima, P.C., (1993), "Um Sistema de Programação Finita Baseado em Lógica Nebulosa", Tese de Doutorado, Faculdade de Engenharia Mecânica, UNICAMP.

Mason, A.J. e Anderson, E.J., (1991), "Minimizing Flow Time on a Single Machine with Job Classes and Setup Time", *Naval Research Logistics*, Vol. 38, pp. 333-350.

Ogbu, F.A. e Smith, D.K., (1990), "The Aplication of the Simulated Annealing Algorithm to the Solution of the $n/m/C_{max}$ Flowshop Problem", *Computers Operations Research*, Vol. 17, Nº 3, pp. 243-253.

Panwalkar, S.S. e Rajagopalan, R., (1992), "Single-Machine Sequencing with Controllable Processing Times", *European Journal of Operational Research*, Vol. 59, pp. 298-302.

Pinedo, M., (1995), *Scheduling. Theory, Algorithms, and Systems*, Prentice Hall.

Pureza, V.M.M., (1990), "Problemas de Roteamento de Veículos Via Metaheurística Tabu", Tese de Mestrado, Faculdade de Engenharia Elétrica, UNICAMP.

Sarin, S.C., Erel, E. e Steiner, G., (1991), "Sequencing Jobs on a Single Machine with a Common Due Date and Stochastic Processing Times", *European Journal of Operational Research*, Vol. 51, pp. 188-198.

Sen, T. e Gupta, S.K., (1984), "A State-of-Art Survey of Static Scheduling Research Involving Due Dates", *OMEGA*, Vol. 12, Nº 1, pp. 63-76.

Solomon, M.M., (1987), "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints", *Operations Research*, Vol. 35, Nº 2, pp. 254-265.

Solomon, M.M. e Desrosiers, J., (1988), "Time Window Constrained Routing and Scheduling Problems", *Transportation Science*, Vol. 22, Nº 1, pp. 1-13.

Szwarc, W., (1989), "Single-Machine Scheduling to Minimize Absolute Deviation of Completion Times from a Common Due Date", *Naval Research Logistics*, Vol. 36, Nº 5, pp. 663-673.

Taillard, E., (1990), "Robust Taboo Search for the Quadratic Assignment Problem", *Département de Mathématiques, École Polytechnique Fédérale de Lausanne*, Working Paper ORWP 90/10, Suiça.

Widmer, M. e Hertz, A., (1987), "A New Approach for Solving the Flow Shop Sequencing Problem", *Département de Mathématiques - Chaire de Recherche Opérationnelle*, CH-1015, Lausanne, Suiça.

Woodruff, D.L e Spearman, M.L., (1992), "Sequencing and Batching for Two Classes of Jobs with Deadlines and Setup Times", *Production and Operations Management*, Vol. 1, Nº 1, pp. 87-102.

Zdrazalk, S., (1991), "Approximation Algorithms for Single-Machine Sequencing with Delivery Times and Unit Batch Set-up Times", *European Journal of Operational Research*, Vol.51, pp. 199-209.

Zheng, W.X., Nagasawa, H. e Nishiyama, N., (1993), "Single-Machine Scheduling for Minimizing Total Cost with Identical, Asymmetrical Earliness and Tardiness Penalties", *International Journal of Production Research*, Vol. 31, Nº 7, pp. 1611-1620.