NERONE MARQUES DE BRITO JÚNIOR

UMA APLICAÇÃO DO SIMULATED ANNEALING (SA) PARA PROGRAMAÇÃO DE ORDENS NO CHÃO DE FÁBRICA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Mestre em Engenharia.

São Paulo

2001

NERONE MARQUES DE BRITO JÚNIOR

UMA APLICAÇÃO DO SIMULATED ANNEALING (SA) PARA PROGRAMAÇÃO DE ORDENS NO CHÃO DE FÁBRICA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Mestre em Engenharia.

Área de Concentração:
Engenharia Mecânica

Orientador:
Prof. Dr. Marcos R. P. Barretto

São Paulo

2001

AGRADECIMENTOS

Ao meu pai e minha mãe, que durante boa parte de suas vidas, não pouparam esforços para poder dar a melhor educação possível a seus filhos.

Às minhas irmãs e meu cunhado pelo apoio moral durante este período.

À Elen Paula ("a baixinha") pela sua compreensão, ajuda e apoio nos últimos dois anos.

Aos meus amigos do mestrado, especialmente Leonardo Wellisch de Barros Barreto pelas contribuições e estímulos ao presente trabalho.

Ao meu orientador, Marcos Ribeiro Pereira Barretto, por sua paciência e orientação ao longo do tempo.

À Universidade de São Paulo, representada pela Escola Politécnica da USP, pelas oportunidades dadas na expansão do conhecimento científico do nosso país.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

RESUMO

ABSTRACT

1 INTRODUÇÃO 1

1.1 Objetivos 1

1.2 Justificativas do Trabalho 2

1.3 Organização do Trabalho 4

2 PLANEJAMENTO ESTRATÉGICO E PLANEJAMENTO DA PRODUÇÃO 6

2.1 Planejamento Estratégico e os Sistemas Produtivos 6

2.2 Planejamento Agregado de Produção 10

2.3 Modelos de Planejamento Agregado da Produção 13

3 PROGRAMAÇÃO DA PRODUÇÃO 14

3.1 Definições 14

3.2 Termos utilizados 16

3.3 Definição do problema de "scheduling" e principais suposições 17

3.4 Classificação dos Problemas de "scheduling" 19

3.5 Objetivo ou Desempenho de um Programa da Produção 22

3.6 Características Especiais do Problema de "scheduling" 24

3.7 Natureza Combinatória do Problema 26

3.8 Abordagens para a solução do problema de "scheduling" 26

3.8.1 Regras de Fila 26
3.8.2 Métodos Otimizantes 29
3.8.2.1 Programação Linear 29
3.8.2.2 Algoritmo de Johnson 30
3.8.2.3 Programação Dinâmica 32
3.8.2.4 "Branch and Bound" 32
3.8.2.5 "Beam Search" (BS) 34
3.9 Métodos Heurísticos - Busca na Vizinhança 36
3.9.1 Procedimento Geral de Busca na Vizinhança 38
3.9.2 "Tabu Search" (TS) 38
3.9.3 Algoritmos Genéticos (GA) 41
3.9.4 "Simulated Annealing" (SA) 44
3.9.5 Heurísticas Baseadas em Gargalos 52
3.9.5.1 OPT 52
3.9.5.2 "Shifiting Bottleneck" 53
3.9.5.3 "Bottleneck Dynamics" 55
4 MODELAGEM ADOTADA 57
4.1 Introdução 57
4.2 O problema do planejamento do arranjo físico ou problema de "layout" 57
4.2.1 Definições 57
4.2.2 Formulação e Modelagem 60
4.2.2.1 Função Objetivo 60
4.2.2.2 Modelagem geométrica das atividades 61
4.2.2.3 Modelagem geométrica do contorno 61

4.2.2.4 Medida de distância....62
4.2.2.5 Método e Técnica de Resolução....62
4.3 A analogia entre o "layout" e o "scheduling"....64
4.3.1 A analogia e sua formulação....64
4.3.2 Modelo de Otimização Adotado....72
5 ESTUDOS DE CASOS....75
5.1 Resultados para problemas "job-shop" e "flow-shop" com objetivo de mínimo "makespan"....77
5.2 Problemas de Uma máquina com objetivo de mínimo atraso ponderado (T_{wt}) .84
5.3 Problemas de Duas máquinas do tipo "Flow Shop" com objetivo de atraso total (T_t) 88
5.4 Discussão dos Resultados....92
5.4.1 Aplicação da metodologia a diferentes configurações de problema....92
5.4.2 Escolha da semente inicial....92
5.4.3 Convergência do algoritmo e suas soluções....93
5.4.4 Escolha dos parâmetros....93
6 CONCLUSÃO....95
6.1 Conclusões e contribuições do trabalho....95
6.2 Sugestões para trabalhos futuros....96
7 BIBLIOGRAFIA....97

LISTA DE TABELAS

Tabela 2.1 Fatores de diferenciação das 3 categorias decisórias (extraído de HAX (1984)) 7

Tabela 3.1 Principais Regras Simples 27

Tabela 3.2 Regras e resultados aplicados ao problema de programação de uma máquina , extraído de HAX (1984) 28

Tabela 5.1: Dados do problema 1 77

Tabela 5.2 : Parâmetros do problema 1 77

Tabela 5.3: Solução detalhada do problema 1 78

Tabela 5.4: Dados do problema 2 79

Tabela 5.5 : Parâmetros do problema 2 79

Tabela 5.6: Solução detalhada do problema 2 80

Tabela 5.7: Dados do problema 3 81

Tabela 5.8 : Parâmetros do problema 3 82

Tabela 5.9 Solução detalhada do problema 3 83

Tabela 5.10 Resultados para problemas de uma máquina com objetivo T_{wt} 84

Tabela 5.11 Dados do problema 4 85

Tabela 5.12 Parâmetros do problema 4 85

Tabela 5.13 Dados do problema 5 86

Tabela 5.14 Parâmetros do problema 5 86

Tabela 5.15 Dados do problema 6 87

Tabela 5.16 Parâmetros do problema 6 88

Tabela 5.17 Resumo das soluções para problemas "Flow Shop" de duas máquinas 89

Tabela 5.18 Dados do problema 7 89

Tabela 5.19 Parâmetros do problema 7 90

Tabela 5.20 Dados do problema 8 91

Tabela 5.21 Parâmetros do problema 8 91

LISTA DE FIGURAS

Figura 2.1 Exemplos de Produto, Processo e Programação de Produção - Abordagem Estruturada adaptada de HILL; JONES 1992 9

Figura 2.2 - Classificação dos métodos de soluções e formulações para Planejamento Agregado da Produção segundo BUFFA (1979) 12

Figura 4.1 Exemplo de Cartas DE PARA 59

Figura 4.2 Exemplo de Cartas de Interligações Preferenciais 59

Figura 4.3 Analogia entre "layout" e "scheduling" 65

Figura 4.4 Cálculo de distância para operações de ordens distintas e mesma máquina 66

Figura 4.5 Exemplo de sobreposição de atividades 68

Figura 5.1 Gantt resultante do problema 1 78

Figura 5.2 Gantt resultante do problema 2 80

Figura 5.3 Gantt resultante do problema 3 84

Figura 5.4 Gráfico Gantt para problema 4 85

Figura 5.5 Gantt resultante do problema 5 87

Figura 5.6 Gráfico Gantt resultante do problema 6 88

Figura 5.7 Gráfico Gantt resultante do problema 7 90

Figura 5.8 Gráfico Gantt resultante do problema 8 91

LISTA DE SÍMBOLOS

p_{ij} – tempo de processamento da operação j da ordem i

r_{ij} – tempo ou data de chegada da operação j da ordem i

d_{ij} – data ou tempo de entrega da operação j da ordem i

C_{ij} – tempo final de execução da operação j da ordem i

O_i – Ordens de serviço i

M_j – máquina j

i – índice da ordem de produção ou serviço

j – índice da operação

k – índice de máquina

n_i – número de operações da ordem i

p_{jk} – tempo de processamento da operação j na máquina k

u – pesos para as diferentes variáveis de produção

F_j – quantidade de tempo que a atividade (ordem ou operação) j fica no sistema:

L_i – quantidade de tempo em que ordem i excede ou não a data de entrega:

T_i – atraso da ordem i

E_i – adiantamento da ordem i

C_{max} – tempo máximo de fluxo

F_{wt} – tempo de fluxo ponderado

L_{wt} – desvio ponderado

T_{wt} – atraso ponderado

E_{wt} – adiantamento ponderado

TE_{wt} – adiantamento e atraso ponderados

F_{max} – tempo máximo de fluxo

L_{max} – tempo máximo de desvio

T_{max} – atraso máximo

N_{wt} – número ponderado de ordens em atraso

Z – função objetivo

C_{ik} – tempos de conclusão das ordens i na máquina k

H – número positivo inteiro

χ_{ipk} – variável de precedência

α – "beam factor"

β – "branching factor"

P – probabilidade de aceitação

PA(1) – probabilidade de aceitação inicial

fp – fator de descréscimo da probabilidade de aceitação

e – número de estágios de execução do SA

T_0 – temperatura inicial

T_f – temperatura final

γ – constante de resfriamento ($0<\gamma<1$)

η – constante de decréscimo de temperatura de Johnston ($\eta>1$)

λ – constante de temperatura de Johnson

S – solução

f_{ij} – medida de iteração (fluxo) entre as atividades i e j

a_{ij} – medida de distância entre as atividades i e j

ρ_{ij} – fator de conversão para custos entre as atividades i e j

v_t – são os custos fixos de localização de cada atividade t

b_{ij} – custo fixo de posicionar atividade i na localização j

v_{ik} – custo unitário da medida de iteração entre as atividades i e k

f_{ik} – medida de iteração entre as atividades i e k (fluxo de materiais)

a_{jl} – distância ou afastamento entre as localizações j e l

μ_{ij} – função que toma valor 1 se atividade i for posicionada na posição j ou 0 caso contrário

O_{ijk} – operação j, da ordem i, a ser processada na máquina k

x_{ijk} – coordenada x (abcissa) da ordem O_{ijk}

y_{ijk} – coordenada y (ordenada) da ordem O_{ijk}

h_{ijk}– altura da ordem O_{ijk}

w_{ijk} – largura da ordem O_{ijk}

f_{ijpq} – fluxo entre as ordens O_{ij} e O_{pq}

a_{ijpq} – distância entre as ordens O_{ij} e O_{pq}

α_1 – peso do custos de transporte da função objetivo (Z);

N – o número total de atividades ou operações

t(i) – o número total de operações de cada ordem Oi

α_2 – peso da função de penalidade da função objetivo (Z)

I_{ijpq} – função que toma valor maior que zero se as operações O_{ij} e O_{pq} se sobrepõem

E_{ijpq} – diferença entre a soma da média das larguras de duas atividades e o módulo das abcissas do baricentro entre as operações O_{ij} e O_{pq}

α_3 – peso da função de penalidade da seqüência (para problemas de multi–máquinas)

θ_{ijl} – é a função que toma valor 1 se dentro da ordem i, as operações j e l desrespeitam a seqüência tecnológica, o que ocorre se na seqüência da ordem i, a operação j antecede l mas l está posicionada antes de j

α_4 – o peso da função objetivo de "scheduling" ("makespan" ou atraso ponderado);

U_i – é o valor do peso ou prioridade de cada ordem i

RESUMO

Este trabalho apresenta uma proposta de solução do problema de programação de ordens através de uma analogia com os problemas de arranjo de espaço físico (problemas de "layout"). O algoritmo de resolução faz uso de uma heurística denominada LEO ("Linear move and Exchange Optimization" - Otimização por trocas lineares), baseada no algoritmo "simulated annealing" (SA). O SA é um algoritmo bastante utilizado para resolver problemas combinatórios, categoria que inclui os problemas de "scheduling" e "layout".

A modelagem suporta cronogramas de produção com restrições de datas e é adaptável a problemas de uma máquina ou multi-máquina como as do tipo "flow shop" e "job shop". Os resultados alcançados indicam que o algoritmo LEO produz soluções melhores que algumas regras normalmente utilizadas como EDD e SWPT.

ABSTRACT

In this work, a proposal for solving the Scheduling Problem through an analogy to the facility layout problems (FLP) is presented. The scheduling problem is solved using an heuristic algorithm called LEO (Linear move and Exchange Optimization) algorithm. The LEO algorithm is based on Simulated Annealing (SA), an algorithm for solving hard combinatorial optimization problems, as the Scheduling Problem and the FLP. This approach can support constrained schedules with due dates and other goals as well as multi-machine and single machine scheduling problems. Results for some experiments show that the LEO algoritm outperforms rules such as the EDD and SWPT rules.

1 INTRODUÇÃO

1.1 Objetivos

O presente trabalho tem por objetivo o desenvolvimento de um modelo heurístico de busca em vizinhança estendida, segundo classificação proposta por PACHECO; SANTORO (1999), para o problema de "scheduling"[1] de um sistema produtivo do tipo "job–shop" [2] estático determinístico. A heurística é baseada no algoritmo "simulated annealing" (SA). O uso do SA justifica–se pela facilidade de sua implementação e extensa aplicação a problemas de natureza combinatória como o "scheduling".

Para se atingir este objetivo propõe–se um modelo baseado na analogia entre os problemas de "scheduling" de produção e o problema de alocação de unidades produtivas (problema de "layout"), aproveitando–se as similaridades de ambos em quanto à:

- modelagem matemática
- métodos de resolução
- natureza combinatória
- formas de exposição da solução final

Para o sistema de geração dos cronogramas foi desenvolvido um algoritmo, codificado em Visual Basic, que expõe as soluções finais como um gráfico Gantt, uma ferramenta de programação de produção usada na indústrias.

---

[1] A palavra "scheduling" pode ser traduzida como programação de produção ou agendamento da produção [TEDESCHI (1996)]

[2] A palavra "job-shop" não tem um correspondente exato no português, sendo referida em inglês neste trabalho. Sua definição pode ser encontrado na seção 3.4.

A abordagem proposta pretende ser flexível o suficiente para adaptar–se à realidade do chão de fábrica, abrangendo aspectos tais como múltiplos objetivos (com e sem datas de entrega) e incluindo eventos como manutenção preventiva e quebras de máquinas.

1.2 Justificativas do Trabalho

O aumento da competição nos mercados, decorrente do fenômeno da globalização econômica, tem feito com que as empresas reverem seus processos e estratégias. HILL; JONES (1992) destacam, dentre as estratégias genéricas utilizadas pelas empresas neste novo contexto, a estratégia de liderança em custo.

Para competir com sucesso, as organizações têm sido forçadas a:

- atender datas
- reduzir estoques
- reduzir lead–times
- maximizar a utilização de seus recursos.

Dentro desta realidade, as empresas têm procurado novas técnicas de gerenciamento que propiciem às suas operações alcançar as metas expostas acima.

A programação de produção insere–se como uma atividade crítica para o alcance das metas de redução de custos dentro das empresas de manufatura. Porém, nem sempre as empresas têm dado a devida atenção ao problema e apesar do grande esforço acadêmico dedicado a este tema nos últimos 40 anos, pouca aplicação prática tem sido verificada como reportado por MCKAY et al. (1988), MACCARTHY; LIU (1993) e DUDEK et al. (1992). Ainda segundo STOOP; WIERS (1995), a atividade de programação continua sendo um papel predominantemente humano e sujeito às suas limitações.

No meio acadêmico, grande atenção tem sido dada ao problema a partir da década de 50. O interesse acadêmico sobre o tema reside nas seguintes características do problema programação de produção:

- é um problema de formulação matemática;
- possui natureza combinatória complexa e inclui-se na família dos problemas NP completos, de acordo com KARP (1975);
- é um problema comum em diferentes tipos de operações: manufatura, logística e serviços.

Apesar da grande investigação sobre o tema, existe uma grande lacuna entre o teor acadêmico da modelagem, as premissas adotadas por grande parte das pesquisas sobre "scheduling" e a realidade encontrada nas operações. MCKAY et al. (1988) e DUDEK et al. (1992) enumeram algumas características que explicam esta lacuna:

- a existência de múltiplos e conflitantes objetivos nas operações;
- variabilidade do chão de fábrica (quebras, problemas de qualidade, etc.);
- a existência de pouca qualidade nas informações, informações conflitantes ou a falta delas dentro das operações;
- a necessidade do uso da intuição humana para lidar com as lacunas geradas por problemas de falta de informações;
- a existência de inúmeras restrições na programação não levadas em conta nas hipóteses normalmente utilizadas nas pesquisas;
- a natureza dinâmica do problema real em comparação com a abordagem teórica estática;
- a falta de flexibilidade dos algoritmos em lidar com situações distintas das condições assumidas;
- a falta de conhecimento por parte das empresas sobre as pesquisas e teorias relativas à programação de produção;

- a falta de avaliação do impacto de estratégias pobres de programação de produção nas operações;
- a falta de experimentação prática das teorias sobre programação da produção.

Devido às razões expostas acima, as empresas têm aplicado apenas práticas simples auxiliares da programação como JIT, "kanban" no chão de fábrica, e sistemas MRP em um nível mais agregado, transformando a tarefa de programação de produção em programação de necessidades e ignorando variáveis importantes como capacidade de produção. Isto reduz a compatibilidade de objetivos do sistema produtivo com as metas organizacionais. Dentre as aplicações práticas vigentes, destacam–se as regras de prioridade, cuja principal vantagem reside em sua simplicidade e transparência para o programador de produção.

Portanto, a principal motivação deste trabalho é estudar uma nova modelagem do problema de "scheduling" de modo a conseguir atingir três finalidades:

i. propor nova estratégia de modelagem do problemas de "scheduling" de modo a conseguir flexibilidade suficiente para acomodar um número maior de restrições e objetivos do que normalmente se encontra na literatura;
ii. aplicar o algoritmo de otimização Simulated Annealing (SA) ao problema modelado de modo a obter soluções melhores do que as encontradas com os métodos tradicionalmente utilizados nas indústrias;
iii. contornar o problema da dificuldade de compreensão do sistema de programação de produção através da sua modelagem com analogias baseadas nos gráficos Gantt e no problema de "layout".

1.3 Organização do Trabalho

A organização do presente trabalho é a seguinte:

O Capítulo 1 apresenta os objetivos do trabalho, sua motivação e sua organização.

O Capítulo 2 situa brevemente o problema de "scheduling" dentro de um contexto mais amplo, o do Planejamento Estratégico e de Produção. Este capítulo contém referências aos principais modelos planejamento.

O capítulo 3 contém toda a classificação, simbologia, definição do problema de programação da produção. Apresenta também uma revisão sobre os critérios de avaliação do problema mais utilizados na literatura bem como uma revisão das principais abordagens utilizadas para a resolução do problema, focalizando–se particularmente nos problemas relativos à "flow–shop" e "job–shop".

O Capítulo 4 apresenta a modelagem do problema adotado, bem como as analogias entre o problema de "layout" e o problema de "scheduling". O método de resolução do problema é revisado e detalhado.

O Capítulo 5 mostra os estudos de caso efetuados com o algoritmo descrito no capítulo 4. Os experimentos são detalhados e os resultados experimentais são apresentados bem como seus respectivos resultados são discutidos.

O Capítulo 6 conclui o trabalho. Futuras linhas de pesquisa sobre o tema também são apontadas nesta seção.

Finalmente, o Capítulo 7 apresenta toda a bibliografia pesquisada para este trabalho.

2 PLANEJAMENTO ESTRATÉGICO E PLANEJAMENTO DA PRODUÇÃO

2.1 Planejamento Estratégico e os Sistemas Produtivos

Os sistemas de produção objetivam o adequado gerenciamento do fluxo de materiais, desde a compra de matéria–prima até a produção e a entrega dos produtos ao consumidor. Segundo HAX; CANDEA (1984); a natureza destes sistemas é tal que os mesmos estão sujeitos a uma grande variedade de restrições e a inúmeros fatores que contribuem para o seu custo total.

Para se entender adequadamente a dimensão do problema de programação de produção e "scheduling" torna–se necessário contextualizá–lo dentro da realidade das operações, ou seja, entender as características desejáveis à programação de produção dentro das metas de uma organização.

Para identificar tais características, dois modelos são brevemente apresentados a seguir. Maiores detalhes sobre eles podem ser encontrados em HAX; CANDEA (1984).

O primeiro modelo, proposto por ANTHONY (1965), refere–se ao processo de decisão dentro dos sistemas produtivos. Ele o classifica em três grandes categorias:

- Planejamento Estratégico – está centrado em estabelecer objetivos de longo prazo e cursos de ação e recursos para o cumprimento destes objetivos. Isto significa, dentro do contexto dos sistemas produtivos, tomar decisões relativas a investimentos e localização de novas instalações.
- Planejamento de Produção – é o processo de utilização eficiente e eficaz dos recursos, para o atingimento dos objetivos organizacionais. Isto se traduz na adequada alocação de recursos (capacidade instalada, centros de distribuição, mão-de-obra, níveis adequados de estoque, etc.).
- Programação da Produção – Anthony o define "como o processo de garantir que atividades específicas sejam conduzidas eficazmente", ou seja, lida com o dia a

dia das operações com decisões do tipo seqüenciamento de ordens, controle de estoques, etc.

Estes três níveis de processo decisório diferem entre si em vários aspectos. As principais diferenças encontram–se melhor explicitadas na tabela abaixo:

Fator	Planejamento Estratégico	Planejamento da Produção	Programação da Produção
Propósito	Gerenciamento de mudança, aquisição de novos recursos	Utilização de recursos	Execução, avaliação e controle
Instrumentos de Implementação	Políticas, objetivos e investimentos de capital	Orçamento	Procedimentos e relatórios
Horizonte de Planejamento	Longo	Médio	Curto
Escopo	Abrangente, nível corporativo	Médio, nível de fábrica	Reduzido, nível de chão de fábrica
Nível Gerencial Envolvido	Alta Gerência	Média Gerência	Baixa Gerência
Freqüência de Replanejamento	Baixa	Média	Alta
Fonte de Informação	Externa	Externa/Interna	Interna
Nível de Agregação da Informação	Altamente agregada	Moderadamente agregada	Detalhada
Acuracidade requerida	Baixa	Média	Alta
Grau de incerteza	Alto	Médio	Baixo
Grau de Risco	Alto	Médio	Baixo

Tabela 2.1 Fatores de diferenciação das três categorias decisórias [extraído de HAX (1984)]

O segundo modelo refere–se às implicações da estrutura de produto no desenvolvimento dos sistemas de produção. Do ponto de vista geral, os produtos possuem diferentes tecnologias de processo e distintas políticas de posicionamento em termos de mercado e distribuição enquanto as empresas normalmente possuem três atividades logísticas associadas a ele (compra, manufatura e distribuição). Dentro deste

contexto, BUFFA; MILLER (1979) e HAX;CANDEA (1984) classificam os sistemas produtivos em quatro grandes classes:

- Sistemas de estoque puro – são sistemas do tipo atacadista (supermercados, grandes magazines) e caracterizam–se por não terem o processo de manufatura.
- Sistemas de Produção Contínua – são sistemas que se caracterizam por produtos de roteiro fixo, fabricados em grande quantidade e com políticas de carregar estoques. Normalmente se associam a instalações com equipamentos dedicados e dispostos em linha (linha de produção).
- Sistemas de Produção Intermitente – são sistemas que fabricam produtos com roteiros variáveis e envolvem vários centros de máquinas. Este tipo de sistema produtivo pode ainda ser subclassificado em sistemas que produzem para estoque ("closed–shop") ou por pedido ("open–shop").
- Gerenciamento de Projetos – é o caso particular de um sistema intermitente onde cada produto é único e normalmente feito uma única vez.

As duas abordagens apresentadas acima levam a duas conclusões:

- o processo de Planejamento da Produção e Programação da Produção deve ser hierárquico e consistente; e
- as características da estrutura do produto afetam as decisões em relação aos modelos de planejamento e programação de produção que devem ser adotados por cada operação. Esta relação pode ser mais bem entendida na figura 1.1.

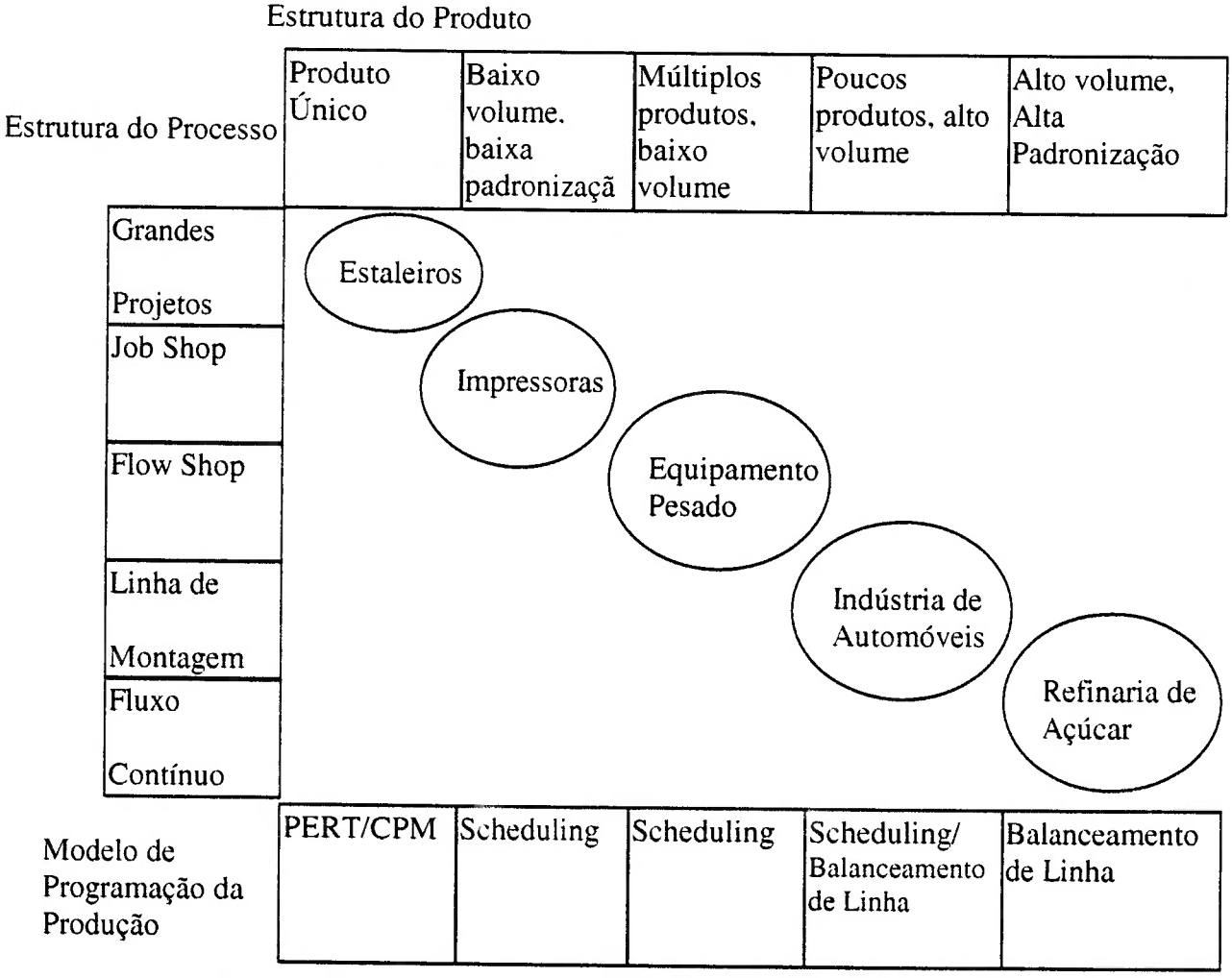

Figura 2.1 Exemplos de Produto, Processo e Programação de Produção – Abordagem Estruturada adaptada de HILL; JONES (1992)

2.2 Planejamento Agregado da Produção

O Planejamento Agregado da Produção corresponde ao segundo nível decisório do modelo hierárquico de ANTHONY (1965). O objetivo do planejamento agregado da produção é a determinação dos níveis de estoque e mão-de-obra capazes de atender às flutuações da demanda. Neste nível de planejamento, os recursos físicos da empresa são assumidos fixos durante o período de planejamento, sendo determinados em nível superior (a fase de planejamento estratégico).

Os modelos de planejamento agregado pressupõem um período de planejamento suficientemente longo (normalmente entre 12 e 18 meses) e um nível agregado de informações, ou seja, os produtos devem ser agregados em famílias, as máquinas em centros de máquinas, etc.

Num modelo de planejamento a variabilidade da demanda pode ser ajustada da seguinte forma:

1. variando–se a força de trabalho através da demissão e contratação;
2. variando–se as horas ou turnos trabalhados admitindo–se horas extras ou ociosidade;
3. variando–se os volumes de produção de modo a acumular estoques quando existe capacidade ociosa;
4. mantendo–se um nível constante de produção, admitindo–se o atraso nas ordens de produção ou faltas , quando possível, nos picos de demanda.

Tais decisões estão associadas aos custos de produção que podem ser agregados em:

- *custos básicos de produção* – são custos diretamente relacionados com o produto (mão-de-obra, matérias-primas, etc.). Podem ser agrupados em custos diretos (diretamente relacionados com cada unidade produzida) e indiretos;

- *custos associados com mudanças na taxa de produção* – são os custos associados à variação da produção para níveis superiores aos praticados. Isto se traduz em variar o volume de horas trabalhadas através de variações no nível de mão-de-obra, o que significa contratações, demissões ou horas extras. Para algumas empresas isto também pode significar subcontratações. Tais custos associam–se na realidade a custos inerentes à admissão e treinamento de novos funcionários e encargos salariais e trabalhistas gerados por horas extras e demissões;
- *custos associados com estoque ou custos de estocagem* – estes custos referem–se aos custos de movimentar estoques (relacionados com custos de armazenagem, controle, espaço imobilizado e custo de oportunidade de capital) e à falta do produto (relacionado com custos de venda perdida, perda de fidelidade do consumidor, perda da imagem do produto, etc.).

Uma vez determinados os objetivos e a natureza dos custos envolvidos num modelo de planejamento agregado da produção, sua formulação dependerá do equilíbrio entre o grau de realismo e acuracidade que se deseja e a viabilidade da sua resolução computacional.

BUFFA; MILLER (1979) apresentam uma classificação entre formulações e técnicas de solução para o problema do Planejamento Agregado da Produção que pode ser mais bem entendido na figura abaixo:

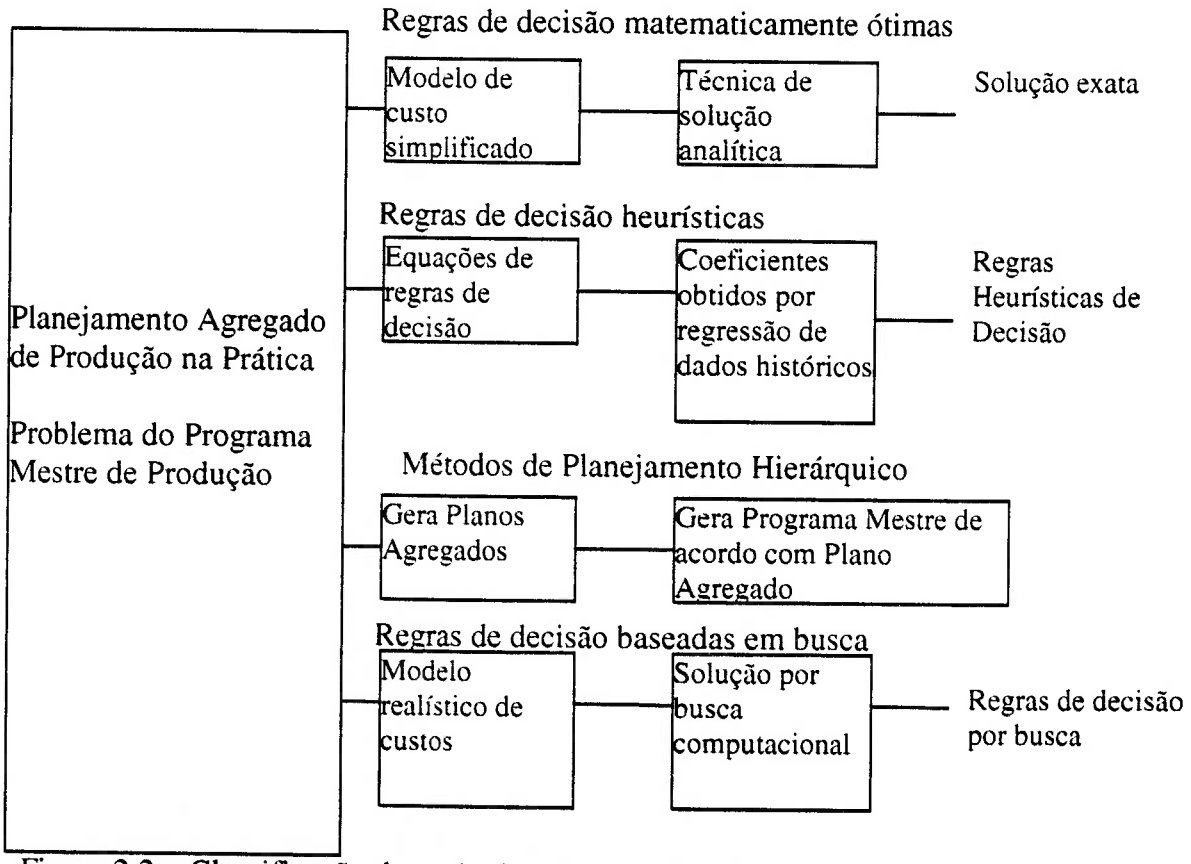

Figura 2.2 – Classificação dos métodos de soluções e formulações para Planejamento Agregado da Produção segundo BUFFA; MILLER (1979)

Na figura acima, Buffa e Miller dividem os modelos de planejamento em quatro tipos, que basicamente se distinguem pelos métodos de formulação (modelos de custos simples ou complexos, baseados em regras de decisão gerencial ou modelos agregados) e pelas soluções geradas (soluções exatas por métodos matemáticos exatos, buscas heurísticas, modelos de agregação/desagregação e regras de decisão).

2.3 Modelos de Planejamento Agregado da Produção

Os modelos de Planejamento Agregado da Produção são baseados em funções objetivo ou funções de custo que representam matematicamente a estrutura de custos (simplificada ou não) da empresa. Portanto, a decisão ótima do modelo será aquela que minimizará os custos modelados por esta função.

A função de custo representa um único objetivo. Para incluir a multiplicidade de objetivos que aparecem nas decisões gerenciais, pode–se utilizar os seguintes artifícios:

- selecionar o objetivo mais importante como o objetivo da função e estabelecer patamares mínimos para os demais através da adição de restrições; ou
- estabelecer uma função objetivo que represente uma ponderação entre todos os objetivos implícitos ao problema

Uma discussão dos modelos e sua respectiva classificação podem ser consultadas em HAX (1984); CANDEA e BUFFA; MILLER (1979). Detalhamento sobre os vários modelos de planejamento agregado da produção bem como respectivos resultados alcançados podem ser encontrados nos trabalhos de HOLT et al. (1960), BYRNE; BAKIR (1994), BOWMAN (1963), NAM; LOGENDRAN (1995), MELLICHAMP; LOVE (1978), JONES (1967) e TAUBERT (1968).

3 PROGRAMAÇÃO DA PRODUÇÃO

3.1 Definições

Segundo BAKER (1974), a programação de produção consiste em distribuir um conjunto de recursos (máquinas) durante um determinado período de tempo de modo a executar um conjunto de atividades com a finalidade de produzir um produto ou bem. O ponto de partida da programação de produção é o planejamento da produção (discutido na seção anterior) que determina os recursos necessários (máquinas, mão de obra), os produtos demandados e quanto deve ser produzido de cada um. Dentro deste contexto, de acordo com MORTON; PENTICO (1993), as decisões relativas à programação de produção encontram–se no nível mais inferior do planejamento hierárquico da produção, necessitando de informações mais detalhadas (como datas de entrega, quantidades por cada tipo de produto) e possuindo um período de planejamento curto, entre 2 e 6 semanas.

O problema da programação de produção surge quando existem uma série de atividades a serem executas com recursos limitados(capacidade, tempo, mão-de-obra) e sujeitas ainda a um conjunto de restrições (datas de entrega, roteiros de processo, etc.).

A solução deste problema é uma ordenação das distintas atividades associadas a um recurso (máquina) ao longo do tempo que satisfaz uma ou várias metas gerenciais pré–determinadas. O produto desta solução pode tomar as mais diversas formas como uma tabela que contém atividades, recursos e datas ou um gráfico Gantt que aponta o tempo utilizado de cada atividade por cada recurso.

Devido à quantidade de variáveis envolvidas no processo de programação, às inúmeras restrições e à multiplicidade de critérios de desempenho, o problema de programação de produção tem atraído o interesse acadêmico.

Para BUFFA; MILLER (1979), HAX;CANDEA (1984), MORTON; PENTICO (1993), BAKER (1974), o problema geral de programação de produção é classificado em quatro tipos distintos de problemas:

"Scheduling" – é o problema de programação de produção propriamente dito. Consiste em ordenar as atividades nos recursos associando–se datas (datas de início e fim).

Seqüenciamento – resume–se a associar uma seqüência ou ordem de atividades a cada recurso. O produto do seqüenciamento será o mesmo do "scheduling" para os casos em que não houver inserção de esperas nos recursos. Tanto o seqüenciamento como o "scheduling" são problemas decorrentes dos sistemas de produção intermitentes que compõem grande parte dos sistemas de manufatura e serão abordados em maiores detalhes no presente trabalho.

Balanceamento de linhas – é a aplicação do problema de programação da produção para o caso de um sistema de produção contínua. Neste caso, o problema de programação consiste em dividir as tarefas de modo a reduzir a diferença entre as quantidades de trabalho alocadas a cada estação. Isto significa:

- obter o menor número de estações de trabalho possível para uma taxa de produção determinada (problema de balanceamento tipo I);
- tendo um número fixo de estações, definir o tempo de ciclo (tempo de trabalho para cada estação) que minimize a diferença de quantidades de trabalho de cada estação (problema de balanceamento tipo II);
- atender os dois objetivos supracitados ao mesmo tempo (problema de balanceamento tipo III).

Um aprofundamento do problema de balanceamento bem como os principais métodos de resolução podem ser encontrados em HAX; CANDEA (1984) e ABREU (1996).

Gerenciamento de Projetos – é um caso especial de produção intermitente, em que cada produto é único e somente executado uma vez. Pelas particularidades deste tipo de produção, a abordagem é distinta em relação ao "scheduling" e ao seqüenciamento. Uma definição do problema, bem como métodos de solução podem ser encontrados em HAX; CANDEA (1984), BAKER (1974) e MORTON; PENTICO (1993).

3.2 Termos utilizados

Para um melhor entendimento do texto a ser apresentado seguem alguns termos normalmente utilizados na literatura e que aparecerão com freqüência no desenvolver do trabalho. A maior parte dos termos utilizados encontram–se no trabalho de CONWAY et al. (1967), BAKER (1974), MORTON; PENTICO (1993), e PINEDO (1995):

- Operação ou Atividade: é uma tarefa elementar a ser executada.
- Ordem de Serviço ou Ordem de Produção: é uma requisição para a fabricação de um produto. Possui um conjunto de operações ou atividades e um roteiro.
- Máquina: uma máquina ou equipamento capaz de executar a operação.
- Roteiro: indica as relações de precedência entre as operações da mesma ordem de serviço.
- Índices i e j: a nomenclatura utilizada nos trabalhos anteriormente citados utiliza o índice i para representar ordens de produção, enquanto o índice j é utilizado para representar as distintas operações que compõem a ordem i

- Tempo de Processamento (p_{ij}): é o tempo necessário para processar uma operação.
- Tempo de chegada (r_{ij}): é o tempo mais breve que uma ordem de serviço é expedida para o chão de fábrica.
- Data de entrega (d_{ij}): é o tempo em que a última operação de uma ordem de serviço deve ser executada.
- Tempo de execução (C_{ij}): é o tempo computado para que a última operação de uma ordem de serviço seja completada.
- Programa de Produção (ou Cronograma de fabricação): é a ordenação de cada uma das operações contidas em todas as ordens de serviço em todas as máquinas disponíveis no chão de fábrica.
- Tempo de preparo da máquina ("setup–time"): é o tempo necessário para preparar a máquina para a operação seguinte a ser processada. Pode ser dependente ou independente da seqüência de operações da máquina. Na maioria dos modelos encontrados na literatura, os tempos de preparo são independentes da seqüência de operações da máquina.
- Seqüência: é a seqüência (ordem) de operações na máquina.

3.3 Definição do problema de "scheduling" e principais suposições

A formulação genérica de um problema de "scheduling" obedece à seguinte forma: Dado um conjunto de n ordens de serviço para serem processadas e m máquinas disponíveis para o seu processamento, onde é necessário um subconjunto $m_r \subset m$ destas máquinas. As ordens de serviço (O_i) estão disponíveis para processamento nas datas (tempos) iniciais r_i e devem, ser completadas até da data de entrega (d_i).O processamento de uma ordem de serviço (O_i) em um máquina M_j é uma operação O_{ij}.

Cada operação O_{ij} gasta um tempo de processamento p_{ij} na máquina ($0<i\leq n$, $0<j \leq m$). Cada operação O_{ij} constitui–se, portanto, num subconjunto do conjunto de operações O_k onde ($0<k\leq i \times j$).

O problema de "scheduling" consiste em encontrar uma alocação de cada operação pertencente às ordens de serviço na escala de tempo das máquinas, com o intuito de atingir um objetivo determinado. Contudo, entre duas operações pode–se inserir um tempo arbitrário de espera, o que torna o universo de soluções para o problema infinito. Com o intuito de reduzir as dimensões do problema e simplificá–lo, várias suposições devem ser feitas. As principais suposições são, segundo CONWAY et al. (1967):

i. qualquer máquina está sempre disponível para a execução de uma operação. Os tempos de quebras e manutenção são desprezados;
ii. ordens de serviço são seqüências estritamente ordenadas de operações sem partições ou montagens (não existe ordem de produção como árvore de montagem);
iii. cada operação pode ser executada por apenas uma única máquina no chão de fábrica;
iv. existe uma única máquina de cada tipo no chão de fábrica;
v. não é permitida preempção, isto é, uma vez que uma operação começou em uma máquina, ela não pode ser interrompida antes de ser processada integralmente;
vi. os tempos de processamento de sucessivas operações pertencentes à mesma ordem de serviço não podem ser sobrepostos, ou seja, uma ordem de serviço pode possuir apenas uma operação sendo executada por vez;
vii. cada máquina pode executar somente uma operação por vez;

Os problemas de "scheduling" que satisfazem as restrições acima são considerados básicos. Contudo, os problemas reais não obedecem algumas destas restrições. Ao

mesmo tempo, encontramos na literatura problemas e soluções nos quais algumas destas restrições são abandonadas para aproximar a modelagem da realidade dos sistemas produtivos. Estes serão mais bem detalhados no decorrer do presente trabalho.

3.4 Classificação dos Problemas de "scheduling"

Dada a abrangência do problema de "scheduling", uma classificação dos diversos tipos de problemas e abordagens faz–se necessária. Utilizaremos aqui as classificações utilizadas por CONWAY et al. (1967) e RAMASESH (1990).

RAMASESH (1990) propõe uma classificação dos problemas de "scheduling" segundo cinco aspectos:

i. Complexidade do Processo

O chão de fábrica pode possuir as seguintes configurações para o problema de "scheduling":

- *Uma única máquina* – caso mais simples em que cada ordem de serviço possui uma única operação
- *Multi máquinas* – caso mais geral, em que existem m máquinas na produção
- *Máquinas paralelas* – caso especial das multi–máquinas, quando existe mais de uma máquina que pode executar a mesma operação. Um chão de fábrica com máquinas paralelas comporta sempre roteiros alternativos para as ordens de serviço.
- *"Flow–shop"* – caso particular do chão de fábrica com multiplicidade de máquinas. Neste caso, as máquinas são dispostas de modo que as ordens possuam um fluxo único de processamento (não há roteiros alternativos). Assim, cada ordem de serviço possui o mesmo número de operações que são processadas nas mesmas máquinas.

- *"Job–shop"* – caso mais geral das multi–máquinas, onde não existe rigidez no fluxo da produção. As ordens de serviço deveriam ter um número de operações menor ou igual ao número de máquinas m.

ii. Natureza da Geração das Requisições

As requisições (demanda) dos produtos levam às seguintes classificações dos ambientes produtivos:

- *"Open-Shop"*: produção para atender à demanda sem acúmulo de estoques. Os roteiros de fabricação para este caso são variáveis.
- *"Closed-Shop"*: a demanda é atendida através de estoque. Os roteiros neste caso são conhecidos e pertencentes a um conjunto fixo.

iii. Natureza dos dados em relação ao tempo

O processo de chegada das ordens de serviço pode ser estático ou dinâmico.

No processo *estático*, todas as ordens de serviço chegam simultaneamente ao chão de fábrica, num período T (que pode ser zero ou não). Num processo *dinâmico*, as ordens de serviço chegam intermitentemente ao processamento. Para este caso, a literatura tem dois tipos de abordagem:

- as ordens chegam num período futuro (caso determinístico), ou
- as ordens chegam continuamente conforme uma distribuição de probabilidades pré–definida.

iv. Natureza das variáveis de produção

O processamento das ordens de serviço será *determinístico* se os tempos de processamento e os tempos de chegada das ordens forem fixos e conhecidos, com alto grau de certeza.

No caso *probabilístico*, os tempos de processamento ou o tempo de chegada das ordens ou ambos, são aleatórios ou descritos por distribuições de probabilidades.

v. Critério de Desempenho

Os critérios de desempenho de um sistema de programação de produção são normalmente descritos através de funções matemáticas que expressam implicitamente uma meta a ser atendida. Estas funções são chamadas funções objetivo. BAKER (1974) agrupou os objetivos em três tipos:

- Utilização eficiente dos recursos,
- Resposta rápida à demanda, e
- Atendimento a datas.

Uma descrição das funções objetivo, bem como sua relação com as metas gerenciais de um ambiente produtivo pode ser vista na seção 3.5

CONWAY et al. (1967) propuseram um modelo de classificação dos problemas de "scheduling" , bastante utilizado na literatura, baseado em quatro tipos de informação, gerando uma notação da forma A/B/C/D que identifica cada problema de "scheduling" individualmente. Tal notação baseia–se em:

"A" – descreve o processo de chegada das ordens. Para o caso estático, A é o número de ordens que chega simultaneamente. Para o caso dinâmico, A descreve a função de probabilidade da chegada das ordens.

"B" – descreve o número de máquinas contidas no chão de fábrica.

"C" – descreve o padrão de fluxo do chão de fábrica. C pode ser:

- "F" – para o caso de uma "flow–shop",
- "R" – para o caso de roteiros aleatórios, e
- "G" – para um caso geral ou fluxo arbitrário. C é omitido no caso de haver uma única máquina.

"D" – descreve o critério de avaliação do programa de produção (função objetivo).

Por exemplo, o problema N/m/F/Cmax significa um problema de N ordens com m máquinas em um "flow–shop" (F), com o objetivo de minimizar o tempo máximo de fluxo ("makespan" – Cmax).

3.5 Objetivo ou Desempenho de um Programa da Produção

As características da demanda e do processo de um sistema produtivo, bem como as forças externas, podem variar muito de uma empresa para outra e exigem diferentes políticas para a determinação de metas para a programação da produção. Alguns exemplos de metas gerenciais podem ser:

- atendimento a datas de entrega,
- máxima utilização da capacidade instalada,
- minimização do estoque em processo,
- minimização da ociosidade das máquinas,
- minimização dos custos operacionais, etc.

Nos problemas de "scheduling", estes objetivos, bem como outros, normalmente são mensurados matematicamente através de uma função objetivo, que representa um conjunto de informações e medidas sobre cada ordem de serviço no chão de fábrica. Segundo BAKER (1974), esta função depende diretamente dos custos relacionados ao tempo em que o conjunto de ordens demora para ser processado.

Para um melhor entendimento sobre as funções objetivo mais utilizadas na literatura, usar–se–ão as seguintes notações extraídas de MORTON; PENTICO (1993):

i – índice indicando a ordem de serviço i

j – índice indicando a operação j

k – índice indicando a máquina k (k=1...m)

n_i – número de operações da ordem i

r_i – data de chegada da ordem i

d_i – data de entrega da ordem i

p_{jk} – tempo de processamento da operação j na máquina k

u_j – pesos para as diferentes variáveis de produção

Definem–se:

C_i – instante em que a ordem i completa seu processamento

F_j – quantidade de tempo que a atividade (ordem ou operação) j fica no sistema:

$F_j = C_j - r_j$

L_i (Desvio) – quantidade de tempo em que ordem i excede ou não a data de entrega:

$L_i = C_i - d_i$

T_i – atraso da ordem i: T_i = máx $\{0, L_i\}$

E_i – adiantamento da ordem i: E_i = máx $\{0, -L_i\}$

Nestas definições, as funções objetivo mais comuns encontradas na literatura envolvem as seguintes minimizações:

- Tempo máximo de execução ("makespan") – este objetivo traduz a meta de máxima utilização do chão de fábrica. $C_{\max} = \max\{C_j\}$
- Tempo de fluxo ponderado – é definido como: $F_{wt} = \sum_j u_j F_j$
- Desvio ponderado – é definido como: $L_{wt} = \sum_j u_j L_j$. Tanto o tempo de fluxo ponderado como o Desvio ponderado são objetivos bastante comuns e suas soluções ótimas são idênticas [MORTON; PENTICO (1993)].
- Atraso ponderado – é definido como: $T_{wt} = \sum_j u_{Tj} T_j$.
- Adiantamento ponderado – é definido como: $E_{wt} = \sum_j u_{Ej} E_j$.

- Adiantamento e Atraso ponderados – é definido como: $TE_{wt} = \sum_j (u_{Tj}T_j + u_{Ej}E_j)$

  – Este objetivo é utilizado em sistemas produtivos em que o cliente não tolera atrasos e a empresa deve manter baixo nível de estoques (o consumidor não aceitará a entrega adiantada, portanto a empresa precisará estocar no caso de um adiantamento).

- Tempo Máximo de Fluxo – é definido como : $F_{max} = max\{F_j\}$

- Tempo Máximo de Desvio – é definido como : $L_{max} = max\{L_j\}$

- Tempo Máximo de Atraso – é definido como : $T_{max} = max\{T_j\}$. O objetivo de minimizar o máximo atraso está relacionado com sistemas produtivos em que o cliente se torna progressivamente mais descontente conforme o atraso aumenta. , De acordo com MORTON; PENTICO (1993), Este objetivo pode ser utilizado para minimização do tempo máximo de fluxo para o caso em que $r_i=d_i$.

- Número ponderado de ordens em atraso : $N_{wt} = \sum_j u_{Nj}\delta(T_j)$,

  onde: $\delta(x) = 1$ se x>0

  $\delta(x) = 0$ se contrário

3.6 Características Especiais do Problema de "scheduling"

Existem algumas características particulares dos problemas de programação que violam as hipóteses apresentadas na seção 2.2.3, mas que pelas restrições adicionais ao problema e limitação do número de soluções possíveis devem ser evitadas. Estas características são:

- Tempos de preparação ("setup") dependentes da seqüência – em alguns sistemas produtivos, o tempo de preparação de uma ordem O_i depende diretamente da ordem O_j antecessora a ela. Este problema foi extensivamente pesquisado para o

caso de uma única máquina, sendo modelado como um problema análogo ao do caixeiro-viajante. Uma descrição detalhada de tal modelagem pode ser encontrada em HAX; CANDEA (1984), e BAKER (1974).

- Cadeia de ordem ou estrutura de produto – algumas vezes ocorrem restrições adicionais nos casos em que uma ordem está associada a uma árvore de estrutura de produto (árvore de montagem), ou seja, a ordem não pode começar a ser processada sem que outra ordem associada ao mesmo produto esteja pronta.
- Preempção – refere–se à possibilidade de interrupção do processamento de uma ordem para a execução de outra ordem, mais prioritária. Uma abordagem normalmente encontrada em problemas que permitem preempção é a preempção unitária.

A preempção unitária consiste em aproximar as datas de chegada e de entrega para inteiros e permitir o processamento da ordem em blocos, ou seja, "quebrar" uma ordem em várias mini–ordens. Com relação à medição do impacto da preempção unitária nos custos da função objetivo, segundo MORTON; PENTICO (1993), existem três abordagens:

- Custo no Final ("cost at end" – CAE) – cada mini–ordem possui a mesma data de entrega (a mesma da ordem original) e somente a última mini–ordem processada incorre no custo.
- Custo Diluído ("spread cost" – SC) – cada mini–ordem possui a mesma data de entrega. O custo final da função objetivo associada à ordem original incorre num custo $1/p_i$ (onde p_i é o tempo de processamento da mini–ordem i)

- Custo Diluído com Datas ("tight spread cost" – TSC) – o custo da função objetivo é diluído da mesma forma que no Custo Diluído, porém as datas de entrega de cada mini–ordem são escalonadas (d_j, d_j–1, d_j–2).

3.7 Natureza Combinatória do Problema

Os problemas de "scheduling" encontram–se na categoria de problemas da pesquisa operacional que são relativamente fáceis de formular mas difíceis de resolver. Uma das razões reside na natureza combinatória e discreta desta classe de problemas o que faz com que seja inaplicável um método de otimização através de diferenciação.

KARP (1975), em seu estudo sobre complexidade computacional demonstra que problemas de "scheduling" pertencem à classe dos problemas NP completos, ou seja, os algoritmos utilizados para sua otimização possuem um tempo de execução que não pode ser limitado por uma função polinomial do tamanho do problema. De fato, a natureza combinatória dos problemas de "scheduling" faz com que o seu tempo de execução possa ser modelado como uma função exponencial do seu tamanho (determinado pelo número de máquinas, número de ordens e número de operações de cada ordem).

3.8 Abordagens para a solução do problema de "scheduling"

3.8.1 Regras de Fila

As primeiras abordagens ao problema de "scheduling" e programação de produção envolviam regras que determinavam a seqüência com que as ordens entrariam em processamento em cada recurso. PANWALKER; ISLANDER (1977) fizeram uma distinção entre 113 regras levantadas, agrupando–as em 5 categorias:

- Regras de Prioridade Simples – são baseadas em informações simples da atividade (como data de entrega, tempo de processamento, etc.) ou do recurso (tamanho da fila);

- Combinação de regras simples;
- Índices de Prioridade com Pesos – são regras simples ou combinação delas com diferentes pesos;
- Regras Heurísticas – envolvem considerações mais complexas como efeitos de roteiros alternativos, carga da máquina ou modelagem da experiência humana;
- Outras Regras – podem ser regras específicas para as características de um determinado tipo de chão de fábrica ou combinação de índices baseados em uma formulação matemática.

As regras mais comuns da literatura [MORTON; PENTICO (1993), HAX; CANDEA (1984), BAKER (1974)] podem ser encontradas na tabela a seguir:

Regra	Sigla
Tempos Mais Curtos de Processamento	SPT
Tempos Ponderados Mais Curtos de Processamento	SWPT
Tempos Mais Curtos de Processamento Remanescentes	SRPT
Tempos Mais Curtos de Processamento Esperados	SEPT
Data de Entrega mais Próxima	EDD
Data de Entrega Ponderada	WEDD
Primeira a chegar primeira a sair	FCFS
Tempo de Processamento mais Longo	LPT

Tabela 3.1 Principais Regras Simples

Entre os extensivos estudos sobre regras encontrados na literatura, alguns resultados ótimos para problemas estáticos determinísticos de uma máquina com funções objetivo regulares são segundo CONWAY et al.(1967), BAKER (1974), HAX; CANDEA (1984) e MORTON (1993):

- para minimizar o tempo de fluxo ponderado, a seqüência baseada nos menores tempos ponderados de processamento (WSPT) é ótima;
- para minimizar o atraso máximo das ordens ou desvio máximo, a seqüência baseada em data de entrega mais próxima (EDD) é ótima.

As abordagens mais recentes de regras de filas estão ligadas à teoria das filas e são aplicadas para problemas de "scheduling" dinâmicos e probabilísticos com em WIEN; CHEVALIER (1992) e TEDESCHI (1996). Nestes casos, com a aplicação de diferentes disciplinas de regras a diferentes recursos, procura–se obter resultados sub–ótimos para um critério de desempenho preestabelecido. Estudos clássicos destas abordagens podem ser encontrados no trabalho de CONWAY et al. (1967) e BAKER (1974). Uma avaliação detalhada de algumas regras heurísticas de "scheduling" pode ser encontrada no trabalho de DANNENBRING (1977).

A tabela 3.2 resume os principais resultados encontrados com respeito à aplicação de regras de "scheduling" para problemas de uma máquina:

Natureza das Ordens	Natureza dos Tempos de Processamento	Critério de Desempenho (Minimizar)	Regra	Conclusões
Estático	Determinístico	Tempo de Fluxo Médio	SPT	Ótimo
		Atraso Máximo, Desvio Máximo	EDD	Ótimo
		Número de atrasos	Algoritmo de Moore[3]*	Ótimo
Dinâmico e Determinístico	Determinístico	Tempo de Fluxo Médio	SPT sem preempção	Ótimo
		Tempo de Fluxo Médio	SRPT com preempção	Ótimo
		Desvio Máximo	EDD com preempção	Ótimo
Dinâmico Probabilístico	Deteminístico	Tempo de Fluxo Médio	SPT sem preempção	Ótimo
		Tempo de Fluxo Médio	SRPT com preempção	Ótimo
	Probabilístico	Tempo de Fluxo Médio	SEPT sem preempção	Ótimo

Tabela 3.2 Regras e resultados aplicados ao problema de programação de uma máquina, extraído de HAX; CANDEA (1984)

[3] (*) Detalhado em HAX; CANDEA (1984) e BAKER (1974)

3.8.2 Métodos Otimizantes

3.8.2.1 Programação Linear

Como descrito anteriormente, apesar da complexidade computacional dos problemas de programação da produção, sua formulação é relativamente simples e podem ser resolvidos através de algoritmos bem conhecidos pela Pesquisa Operacional.

A formulação por Programação Linear de um problema de Programação pode tomar a seguinte forma:

$$\min \; Z = \sum_{i=1}^{N} (C_{ik}) \; (1) \qquad (3.1)$$

sujeito à:

$C_{ik} - p_{ik} \geq C_{ih}$ (i) para $(i,j-1,h) < (i,j,k)$

$C_{qk} - C_{ik} + H(1 - \chi_{iqk}) \geq p_{qrk}$ (ii) $1 \leq i, q \leq n, \; 1 \leq k \leq m$

$C_{ik} - C_{qk} + H(1 - \chi_{iqk}) \geq p_{ijk}$ (iii) $1 \leq i, q \leq n, \; 1 \leq k \leq m$

$C_{ik} \geq 0$ (iv)

$\chi_{ik} = 0$ ou 1 (v)

onde:

C_{ik} são os tempos de conclusão das ordens i na máquina k, variáveis de decisão que determinam o cronograma de fabricação.

H – representa um número positivo inteiro

p_{ij} – são os tempos de processamento

χ_{iqk} é 1 se a ordem i precede a ordem q na máquina k ou 0, caso contrário

A equação 3.1 é a função objetivo para tempo médio de fluxo. A restrição (i) restringe apenas cronogramas possíveis (uma máquina não pode processar duas ordens ao

mesmo tempo). As restrições disjuntivas (ii) e (iii) garantem que o cronograma respeite os roteiros e as restrições (iv) e (v) restringem matematicamente C_{ik} e χ_{iqk}.

As vantagens desta modelagem são a garantia da geração de um cronograma ótimo e a facilidade de implementação computacional. Porém o modelo de programação linear torna–se inviável computacionalmente para problemas mais complexos, pois o número de equações de restrições aumenta consideravelmente assim como a complexidade da resolução (devido às características NP completas do problema).

Uma implementação mais eficiente deste método foi conseguida por GREENBERG (1968), que resolvia o problema linearmente sem as restrições (ii) e (iii). Caso fosse gerado um cronograma não executável, as restrições eram adicionadas e o problema era dividido em partes segundo o algoritmo "branch and bound".

3.8.2.2 Algoritmo de Johnson

Um resultado que influenciou bastante a pesquisa dos problemas de "scheduling" foi o encontrado por JOHNSON (1954). Ele propôs um algoritmo de otimização eficiente para um problema estático determinístico de um flow–shop de 2 máquinas, observando duas importantes propriedades:

- com relação a qualquer função objetivo regular, isto é, uma função dos tempos de finalização das ordens (C_{ij}), e para m=2; somente é necessário considerar soluções de mesma seqüência de ordens nas 2 máquinas;
- com relação ao objetivo de "makespan" para m>=2, somente é necessário considerar soluções de mesma seqüência de ordens nas máquinas m e m–1.

As provas destas propriedades podem ser encontradas em BAKER (1974) e CONWAY et al. (1967). Levando–se em conta as propriedades acima, Johnson apenas considerou as soluções de cronogramas permutados (isto é, os cronogramas gerados pela mesma

seqüência de ordens nas máquinas 1 e 2). A seqüência ótima é encontrada obedecendo–se a seguinte regra:

(Regra de Johnson) – A ordem i precede j na seqüência ótima se min $(p_{i1},p_{j2}) \leq \min(p_{i2},p_{j1})$

O algoritmo de Johnson para otimizar o problema de "flow–shop" de duas máquinas consiste em:

1. encontrar min (p_{i1},p_{i2});
2. se o mínimo tempo de processamento requerer a máquina 1, colocar a ordem associada na primeira posição disponível da seqüência. Ir para o passo 4;
3. se o mínimo tempo de processamento requerer a máquina 2, colocar a ordem associada na última posição disponível da seqüência. Se houver empate, escolher arbitrariamente;
4. remover a ordem associada da lista em consideração e retornar ao passo 1 até que a lista fique vazia.

Os resultados encontrados por Johnson para 2 máquinas foram estendidos para 3 máquinas. Porém, neste caso, a solução ótima somente pode ser encontrada pelo algoritmo caso a máquina 2 nunca seja o gargalo do sistema, conforme demonstrado em BAKER (1974). Neste caso, o problema é transformado num análogo ao de 2 máquinas, cujos tempos de processamento são:

- $p_{i1}^* = p_{i1}+p_{i2}$
- $p_{j2}^* = p_{j2}+p_{j3}$
- $p_{i2}^* = p_{i2}+p_{i3}$
- $p_{j1}^* = p_{j1}+p_{j2}$

3.8.2.3 Programação Dinâmica

De acordo com BAKER (1974) e MORTON; PENTICO (1993), se a função objetivo de um problema de programação possui uma forma aditiva, isto é, a função é um somatório dos tempos de conclusão das ordens, este problema pode ser resolvido por programação dinâmica

Este método consiste em dividir as ordens do problema em subconjuntos menores de soluções (seqüências parciais), começando–se com subconjuntos unitários até a formação de um único conjunto de todas as ordens, o que significa partir o problema em subproblemas mais simples. A cada estágio (para cada valor de tamanho da seqüência parcial) determina–se recursivamente a ordem que deve ir primeiro, levando–se em conta os cálculos do estágio anterior. No último estágio determina–se a ordem que deve ir primeiro na seqüência completa. Então, mantendo–se os valores das ordens que devem ir primeiro a cada estágio, determina–se a seqüência ótima. Uma aplicação deste método foi efetuada por SRINIVISAN (1971).

A principal desvantagem deste método é, de acordo com MORTON; PENTICO (1993), sua complexidade computacional, o que o torna restrito à solução de pequenos problemas.

3.8.2.4 "Branch and Bound"

"Branch and Bound" é uma das técnicas da pesquisa operacional mais utilizadas em otimização para problemas combinatórios complexos. Sua aplicação abrange problemas clássicos de otimização como caixeiro viajante, caminho crítico de uma rede, programação de produção, problemas de alocação quadrática.

O método procura resolver o problema através de uma árvore de decisão. Para tanto, consiste em dois procedimentos: geração de nós ("branching") e limitação da busca ("bounding").

O procedimento de geração de nós parte o problema em subproblemas que possuem as seguintes características:

- são mutuamente exclusivos e exaustivos subproblemas do original;
- são subproblemas parcialmente resolvidos; e
- são subproblemas menores que o original.

O procedimento de limitação da busca calcula um limite mínimo ("lower bound") para cada subproblema gerado no procedimento de geração de nós. Caso num determinado nível de árvore (nó) tenha–se encontrado um valor pior do que uma solução teste inicial ou o limite mínimo, então este nó não precisa mais ser avaliado e conseqüentemente a árvore não se expandirá mais a partir deste nó. Este efeito limita então a extensão da busca.

A solução–teste (parâmetro de comparação inicial dos limites mínimos de cada subproblema) pode ser obtida das seguintes formas:

- através de uma solução heurística apropriada ao problema,
- através de uma busca estendida na árvore, ou
- através de uma função heurística de aproximação.

Os procedimentos de "branch and bound" podem ser classificados em 2 tipos:

i. "best first search" – segue a estratégia de possuir baixos limites mínimos ("lower bounds"), encontrando soluções candidatas boas em toda a árvore decisória. Esta estratégia possui a vantagem de manter um entendimento global do problema e a desvantagem de necessitar de grande espaço de memória para

armazenar todos os nós ativos e maior tempo para identificar o nó candidato à expansão;

ii. "depth first search" – segue a estratégia de encontrar rapidamente uma boa solução, seguindo os limites inferiores até o fim da árvore e então procurar nesta vizinhança ("backtracking"). Necessita manter uma lista bem menor dos nós ativos, porém pode ter desempenho inferior ao método anterior.

Num problema de "scheduling" da produção, a partição do problema consiste em fixar uma seqüência parcial de ordens no começo ou final da seqüência total contida em cada nó. A função de cálculo do limite inferior é normalmente uma função heurística do custo estimado da seqüência total em cada subproblema, como aplicado por LOMINICK (1965) e IGNALL; SCHRAGE (1965). Dentre as aplicações do método ao problema, destacam–se os trabalhos de IGNALL; SCHRAGE (1965) e LOMINICK (1965) para "flow–shop" utilizando a abordagem de "depth first search". Outras implementações mais recentes podem ser encontradas nos trabalhos de BAKER (1984) e CARLIER; PINSON (1989). Este último serve de base para buscas realizadas na heurística "Bottleneck Dynamics", a ser detalhada posteriormente.

3.8.2.5 "Beam Search" (BS)

"Beam Search" (BS) é um método de solução de problemas combinatórios complexos, que consiste em uma exploração mais eficiente (mais rápida) da árvore de solução do problema do que o "branch and bound". Este método é similar a um "branch and bound", possuindo as seguintes características próprias:

- em cada nível da árvore somente são escolhidos os α melhores nós para serem expandidos (α é denominado "*beam factor*" [ou largura do feixe] e os nós escolhidos são denominados "*beam nodes*");

- cada "beam node" pode gerar no máximo β nós filhos (β é conhecido como *"branching factor"*).

Os valores de α e β determinam a relação entre qualidade da solução e rapidez na sua determinação, ou seja, valores altos de α e β aumentam a dificuldade computacional do algoritmo porém aumentam a probabilidade de se encontrar uma solução ótima (por uma maior exploração da árvore), enquanto valores baixos de α e β fazem com que se chegue mais rapidamente a uma solução, porém a sua qualidade pode ser prejudicada.

Dois outros fatores interferem na geração da solução e complexidade computacional do "beam search". São eles:

- método de geração dos nós ou a vizinhança da busca, e
- função de avaliação utilizada para se escolher os nós mais promissores (aqueles que serão expandidos a cada nível).

Este último fator é tratado de forma semelhante à implementação de um "branch and bound", ou seja, a função de avaliação é uma função heurística que procura estimar o custo do programa de produção parcial gerado a cada nó.

Esta metodologia foi primeiramente aplicada ao problema de "scheduling" por FOX (1983) no sistema especialista ISIS. Contudo Fox não se preocupou neste trabalho com questões de otimização.

OW; MORTON (1988) aplicaram o "beam search" ao problema de uma máquina com o objetivo de estudar a minimização do atraso e adiantamento ponderado (TE_{wt}) e ao problema de uma flow–shop com o objetivo de atraso ponderado (T_{wt}). No projeto do algoritmo, Ow e Morton utilizaram o conceito de "filtragem" do "beam search", utilizando uma heurística simples para selecionar dentre os nós de cada nível os melhores candidatos e, a partir daí, utilizaram uma função de avaliação mais detalhada

para escolher os nós mais promissores para serem expandidos. CHANG et al.(1989) aplicaram o "beam search" ao problema de "scheduling" de uma FMS (Flexible Manufacturing System) com o objetivo de minimizar o "*makespan*" (C_{max}), conseguindo resultados melhores que as regras de despacho normalmente utilizadas porém com desempenho computacional sensivelmente inferior.

3.9 Métodos Heurísticos – Busca na Vizinhança

Como explicado na seção 3.7. os problemas de "scheduling" são NP completos.

As técnicas de resolução matemática como programação inteira, programação dinâmica e "branch and bound" sofrem uma limitação computacional para encontrar soluções exatas ótimas para problemas da natureza do "scheduling" de grande dimensão (problemas NP completos). Devido a estas características, pode–se dizer que a aplicação destas técnicas está restrita a problemas de reduzida dimensão.

A busca na vizinhança consiste num procedimento computacional que, diferentemente da programação dinâmica e do "branch and bound", não garante que a solução ótima será encontrada. Porém, utilizando–se propriedades intrínsecas do problema, consegue–se, através deste método, atingir soluções comparáveis com métodos otimizantes em tempos computacionais viáveis.

Dentro do conceito e procedimento geral de busca na vizinhança, podemos enquadrar procedimentos heurísticos conhecidos e utilizados em pesquisa operacional e "scheduling" como "tabu search", algoritmos genéticos e "simulated annealing", que serão mais bem detalhados nas seções subseqüentes.

Os elementos básicos de uma busca na vizinhança são:

a semente do problema - é a solução inicial ou ponto de partida do processo de busca. Esta solução inicial pode ser gerada aleatoriamente ou obtida através de uma boa solução heurística para o problema.

a vizinhança da solução corrente – consiste no mecanismo de geração de novas soluções a partir da solução corrente. É uma etapa crítica do algoritmo, pois a vizinhança pode restringir o processo de busca (vizinhança restringida) afetando a qualidade da solução ou pode ser ampla (vizinhança estendida) o suficiente a ponto de afetar o desempenho computacional.

Dentre as vizinhanças mais utilizadas em literatura podemos citar:

- *"adjacent pairwise interchange"* – consiste em escolher duas ordens adjacentes na fila de uma máquina e trocá–las de posição.
- *"general pairwise interchange"* – consiste em escolher duas ordens quaisquer na fila de uma máquina e intercambiar suas posições.
- *"insertion"* – consiste em mover a última ordem para a primeira posição da fila e as n–1 ordens uma posição para trás.
- *"k–move"* – consiste em escolher k ordens da fila de uma máquina e movê–las uma posição para direita ou uma posição para a esquerda.
- *"general three way interchange"* – semelhante ao "general pairwise interchange" envolvendo três ordens.

critério de seleção – é o processo de seleção da próxima semente (isto é a melhor solução). Existem vários critérios que podem ser utilizados. Alguns encontrados na literatura, estão listados abaixo:

- escolher como nova(s) semente(s) aquela(s) que melhorarem a função objetivo,
- avaliar todas as soluções na vizinhança e escolher a que apresenta melhor desempenho,
- utilizar uma heurística para avaliar potenciais sementes.

critério de terminação ou critério de parada – é o processo decisório de finalização da busca. Os critérios mais comuns são:

- parar quando toda a vizinhança foi avaliada e não houve melhoria na função,
- parar quando houver apenas incrementos ínfimos na função objetivo,
- parar quando se atingir uma limite máximo de tempo de computação.

3.9.1 Procedimento Geral de Busca na Vizinhança

i. Obter uma solução inicial, isto é, a semente da busca.

ii. Se o critério de parada ocorrer, parar; senão continuar.

iii. Escolher e gerar uma vizinhança para cada semente.

iv. Escolher uma estratégia de avaliação de cada vizinhança (quanto da mesma vai ser avaliada, como vai ser avaliada [através da função objetivo ou utilizando–se uma heurística]).

v. Após avaliar a vizinhança, escolher a(s) nova(s) semente(s).

vi. Retornar ao passo ii.

3.9.2 "Tabu Search" (TS)

"Tabu Search" (TS) é um método de busca em vizinhança que procura contornar o problema de "estagnação" em um mínimo local através da seguinte estratégia de diversificação:

- a cada iteração, um conjunto de movimentos de busca ou gerações de vizinhança é efetuado e a melhor solução encontrada é armazenada;
- a solução atual é armazenada em uma lista tipo "PEPS" (primeira a entrar, primeira a sair) e, enquanto esta solução está presente nesta lista (isto é, a solução é denominada "tabu"), o procedimento não pode retornar a esta solução, o que força a busca a se diversificar.

Este método foi primeiramente proposto por GLOVER (1990) e foi aplicado com sucesso a diversos problemas na pesquisa operacional. Aplicações ao problema de "scheduling" podem ser encontradas em BARNES; LAGUNA (1993) (máquinas paralelas com objetivo de tempo ponderado de fluxo), SKORIN; VAKHARIA (1993) ("flow–shop" de células de manufatura com objetivo de "makespan"), DELL'AMICO; TRUBIAN (1991) ("job–shop" com objetivo de "makespan"); PUNNEN; ANEJA (1993) ("flow–shop" com objetivo de "makespan")

No projeto de um algoritmo "tabu search" aplicado ao problema de programação de produção, devem ser levados em conta os seguintes parâmetros:

- solução inicial – em geral é utilizada uma heurística aplicada às características do problema, particularlmente às relativas ao tipo de fluxo produtivo e função objetivo;
- tamanho da lista "tabu" – é uma das questões críticas do algoritmo e segue duas estratégias:
    i. listas de tamanho fixo – define–se um tamanho máximo para a lista que é mantido constante durante toda a execução. GLOVER (1990) sugere uma lista de tamanho 7.
    ii. listas de tamanho variável – segundo SKORIN; VAKHARIA (1993), há indícios de que implementações com listas variáveis (que modificam seu tamanho durante a execução) produzem melhores resultados. Estes autores implementaram a seguinte estratégia para o problema de uma "flow–line" de uma célula de manufatura.
    - se não houver melhoria na função objetivo durante um certo número de iterações, então reduz–se o tamanho da lista para intensificar a busca na vizinhança em questão;

- se persistir um acréscimo no objetivo após este procedimento então a lista é aumentada para diversificar–se a busca em outras regiões próximas à vizinhança em questão.

Outra estratégia, implementada por DELL AMICO; TRUBIAN (1991) foi esta:

- enquanto houver melhora na função objetivo, gradualmente decrescer a lista "tabu" até um limite mínimo;
- enquanto houver piora na função objetivo, gradualmente aumentar o tamanho da lista "tabu" até um limite máximo.

- critério de aspiração – o critério de aspiração é a regra utilizada para sobrepor–se ao status "tabu" do movimento (não se pode retornar a uma solução marcada como "tabu"). A regra mais utilizada é aquela que permite aceitar movimentos classificados como tabu, caso estes melhorem a função objetivo.
- elementos de memória de longo prazo – referem–se à criação de atributos que permitam explorar regiões na vizinhança antes pouco ou nunca exploradas ou intensificar a busca em regiões da vizinhança "mais promissoras". SKORIN; VAKHARIA (1993) implementaram uma matriz de freqüência de posição das ordens numa seqüência e, a partir desta, construíram seqüências de mínimas e máximas freqüências. A busca recomeçada com as primeiras explorava regiões da vizinhança pouco visitadas, enquanto a busca recomeçada com as últimas explorava regiões mais "promissoras".
- Resultados do "Tabu Search" (TS) aplicados ao problema de "scheduling": KAPOV; VAKHARIA (1993) encontraram os seguintes resultados em comparação com o "simulated annealing" (SA – detalhado na seção 3.9.4) para problemas de programação de uma "flow-line" com objetivo de minimizar o tempo de execução:

- para problemas simples, ambos os algoritmos atingiram o ótimo sendo o "tabu search" mas eficiente em relação ao tempo de execução;
- para uma classe de problemas intermediários, o SA apresentou resultados melhores que o TS porém em tempo computacional sensivelmente maior;
- para problemas mais complexos, o TS apresentou melhores e mais eficientes soluções que o SA.

BARNES; LAGUNA (1993) estudaram os problemas de multi–máquinas (M=2 a 4, N=20 a 30) com o objetivo de minimizar o tempo ponderado de fluxo e encontraram resultados comparáveis a uma implementação de "branch and bound" mas com tempos de execução menores.

MORTON; PENTICO (1993) fez uma comparação entre a implementação do TS feita por DELL'AMICO; TRUBIAN (1991) e a heurística "shifting bottleneck" (detalhada na seção 3.9.8). Esta implementação encontrou valores que diferiam menos de 10% de uma aproximação do ótimo ("lower bound") para problemas maiores (15 máquinas e 20 ordens) e diferenças menores que 0,2% em relação ao ótimo conhecido para problemas de reduzida dimensão (10 máquinas e 10 ordens).

3.9.3 Algoritmos Genéticos (GA)

Os algoritmos genéticos (GA) são técnicas que podem ser utilizadas para otimização com busca estocástica com base na analogia entre o processo evolucionário biológico e processos de busca e otimização.

Num algoritmo genético, as soluções são codificadas como um cromossomo, geralmente uma "string" composta de genes que representam toda a informação sobre o problema. Os cromossomos, ou seja, as soluções encontradas no problema são alterados ou combinados entre si através dos processos de reprodução, "crossover" e mutação. As novas soluções (cromossomos) oriundas deste processo ou seja a nova geração tendem

a apresentar na média melhores objetivos, pois o algoritmo age de modo a simular o processo de evolução biológica da sobrevivência dos mais aptos. Uma maior abrangência sobre os algoritmos genéticos e algumas de suas aplicações podem ser encontradas em GOLDBERG (1989).

Os seguintes passos são necessários para implementar um algoritmo genético ao problema de "scheduling":

i. um esquema de codificação das soluções em cromossomos e genes. Aqui, duas abordagens podem ser utilizadas:
   - a codificação da seqüência em uma "string", em que a posição relativa indica a posição na fila da máquina e, o seu valor, o indexador que identifica a ordem, estratégia esta implementada por REEVES (1995);
   - a codificação de uma seqüência em que a posição relativa de um gene no cromossomo representa o índice da ordem, enquanto o seu valor representa a prioridade desta ordem na fila de uma máquina. Esta abordagem foi utilizada nos trabalhos de KIM (1996) e DELLA CROCE et al. (1992).

ii. População Inicial – normalmente utilizam–se duas abordagens
   - escolha aleatória da população, usada por KIM (1996);
   - geração de sementes (cromossomos) da população inicial utilizando–se heurísticas ou otimização local, estratégia usada por REEVES (1995).

iii. Função Objetivo e Mecanismo de Seleção– os algoritmos genéticos foram idealizados para problemas de maximização. É necessário, portanto, adaptar a função de avaliação de cada cromossomo ao problema de minimização do "scheduling". A abordagem mais utilizada é a de parametrizar os valores da função objetivo de cada membro da população em torno de uma distribuição de

probabilidades nas quais os cromossomos que apresentem menores valores da função objetivo tenham maior probabilidade de serem escolhidos.

iv. Operadores da Produção da Nova Geração:

- Tamanho da nova população: normalmente é um parâmetro a ser estudado conforme o problema. DELLA CROCE et al. (1992) optaram por manter a população constante fazendo o número de nascimentos igual ao número de mortes.
- "Crossover" - o "crossover" refere–se à combinação de dois cromossomos em que parte de um é transcrita para o outro e vice–versa. A implementação de um "crossover" consiste:
  - na escolha aleatória de 2 cromossomos de uma população,
  - na escolha de uma posição relativa nos dois cromossomos,
  - a troca da porção dos cromossomos escolhidos após a posição relativa determinada,
  - na de resolução de quaisquer paradoxos eventualmente criados.
- Mutação - a mutação refere–se à mudança aleatória de um determinado gene. Na implementação do algoritmo, a mutação ocorre de acordo com uma determinada probabilidade, a taxa de mutação.
- Ambos os operadores ("crossover" e "mutação") são os responsáveis pela criação da nova geração ("offspring") e pela diversificação das soluções ou rearranjos na população que são análogos aos processos de intensificação da busca na vizinhança.

v. Resultados dos Algoritmos Genéticos (GA)

Alguns estudos executados por REEVES (1995), DELLA CROCE et al. (1992) e KIM (1996) comparando os algoritmos genéticos (GA) com outros métodos de otimização

para problemas de "scheduling" ("tabu search", "simulated annealing") ou métodos heurísticos ("shifting bottleneck", detalhado na seção 3.9.5.2), chegaram a resultados comparáveis com os métodos supracitados, mas com tempos de execução sensivelmente maiores [DELLA CROCE (1992)] ou no mínimo de mesma ordem como em REEVES (1995). KIM (1996) encontrou melhores resultados aplicando SA que GA. Estes resultados, contudo, não chegam a ser conclusivos pois nenhum dos pesquisadores chegou a dimensionar os parâmetros do GA de modo que seu desempenho apresentasse um melhor relação entre qualidade da solução e tempo de execução.

3.9.4 "Simulated Annealing" (SA)

"Simulated Annealing" (SA) é um algoritmo de otimização combinatória que vem sendo aplicado com sucesso a diversos problemas de pesquisa operacional [(ZEGORDI et al. (1995), MITTHENTHAL et al. (1996), MAMALIS; MALAGARDIS (1996), VAN LAARHOVEN et al. (1992), WERNER (1993), RAGHU; RAJENDRAN (1995), BRUSCO; JACOBS (1993), KIM (1996), WRIGHT (1989), THOMPSON (1996), ABRAMSOM (1991), EGLESE; RAND (1987), WILHELM; WARD (1987), JAJODIA et al. (1991), KOULAMAS et al. (1994), CHWIF (1994), JOHNSON et al. (1989)]. Uma extensiva biografia das aplicações do SA a problemas de otimização em pesquisa operacional pode ser encontrada em KOULAMAS et al. (1994) e BRANDIMARTE (1992).

O algoritmo foi primeiramente desenvolvido por METROPOLIS (1953) para simular a evolução de um sólido ao seu estado energético de equilíbrio térmico. KIRKPATRICK et al. (1983) e, independentemente CERNY (1985), desenvolveram a analogia entre a minimização de uma função de custo de um problema de otimização combinatória e o resfriamento de um sólido utilizando o processo descrito por Metropolis. Através da

execução do algoritmo de Metropolis em uma seqüência de temperaturas decrescentes originou–se o "Simulated Annealing" (SA).

O SA pode ser visto como um método que combina busca local e melhorias iterativas com processos probabilísticos que atuam no sentido de evitar com que o algoritmo fique estagnado em um ótimo local. Isto é conseguido através da aceitação de soluções que piorem o valor da função objetivo de acordo com uma probabilidade que se altera ao longo de sua execução.

O projeto de um algoritmo SA requer os mesmos parâmetros específicos de cada problema que os demais algoritmos de busca na vizinhança a saber:

- a delimitação de uma vizinhança para o problema – CHEH et al. (1991) concluíram que vizinhanças menores produzem melhores soluções que vizinhanças muito grandes;
- um mecanismo de perturbação para a geração de novas soluções
- uma função objetivo para avaliar as soluções;
- solução inicial – JOHNSON et al. (1989) concluíram após uma série de experimentos, que se a estrutura do problema for tal que o SA não gere soluções que incorporem características especiais do problema, então é preferível a utilização de heurísticas para gerar soluções iniciais e guiar o curso do algoritmo. Caso contrário não há diferença prática em começar com sementes aleatórias ou geradas por heurísticas. Contudo, POTTS; VAN WASSENHOVE (1991) afirmam que se o SA efetuar um número relativamente pequeno de iterações, há ganho em iniciar com uma boa solução guiada por uma heurística adaptada às características do problema.

Além disso, o SA necessita da definição de alguns parâmetros:

i. Probabilidade de aceitação

A probabilidade de aceitação controla, a cada estágio do algoritmo, a possibilidade de diversificação através da incorporação de soluções que piorem o objetivo. No método original, a probabilidade é uma função exponencial da forma $e^{-\Delta/T}$, onde:

- Δ é a diferença entre o custo da solução atual e a melhor solução encontrada, e
- T é um outro parâmetro, conhecido como temperatura (em analogia ao resfriamento dos sólidos).

Nota–se que a aceitação é automática para soluções que melhorem o objetivo (pois a parcela Δ torna–se positiva) e que a probabilidade dependerá da natureza da função de decréscimo da temperatura (melhor detalhada no ponto a seguir). Contudo, há implementações onde existem as seguintes variações:

- Alternância da probabilidade de aceitação [KOULAMAS et al. (1994)];
- Variação da própria probabilidade de aceitação ao invés da temperatura no curso do algoritmo – OGBU; SMITH (1990) utilizaram uma função de decréscimo de probabilidade de aceitação da forma $P=PA(1)\ fp^{e-1}$, onde:
  - P – probabilidade de aceitação
  - PA(1) – probabilidade de aceitação inicial (determinada empiricamente)
  - fp – fator de descréscimo da probabilidade de aceitação (<1)
  - e – número de estágios de execução do SA
- Alteração da natureza da função de probabilidade - No trabalho de POTTS; VAN WASSENHOVE (1991), em que o SA foi aplicado ao problema de "scheduling" do tipo "flow-shop" , utilizou–se a seguinte função de probabilidade de aceitação: $P_i=aP_{i-1}$ onde $a=(P_f/P_o)^{1/(L-1)}$ sendo P_i a probabilidade de aceitação e os parâmetros P_f, P_o e L determinados experimentalmente.

- Aproximações à exponenciação - JOHNSON et al. (1989) utilizaram uma aproximação de exponencial com significativa redução do tempo de computação e efeito mínimo sobre a qualidade da solução.

ii. Evolução do Resfriamento[4]

A evolução do resfriamento refere–se à redução da probabilidade de aceitação durante a execução do algoritmo. Na implementação original, isto significa tomar decisões quanto à:

- "Temperatura" Inicial - significa determinar uma probabilidade inicial de aceitação das soluções. As características das soluções obtidas pelo algoritmo vão depender deste parâmetro inicial. Extensivos experimentos conduzidos por JOHNSON et al. (1989) concluíram que começar em "temperaturas" altas diminui a performance computacional do algoritmo sem ganho qualitativo na solução final enquanto começar em "temperaturas" baixas deteriora a qualidade da solução final obtida. Nas implementações do SA, a temperatura inicial pode ser escolhida por meio de formas distintas:
  - empiricamente, através da fixação da probabilidade de aceitação inicial: JOHNSON et al. (1989) fixaram a probabilidade de aceitação inicial em 40% (uma valor determinado experimentalmente) e determinou a partir dela, o nível de "temperatura" inicial onde a porcentagem de movimentos aceitos no algoritmo fosse aproximadamente 40%.
  - método de CONNOLY (1990) - determina as variações máxima e mínima do custo para uma seqüência de 50 movimentos aleatórios ("pairwise interchange"). Determinado–se fmin (valor mínimo) e fmax (valor máximo) da

---

[4] Em inglês, "*annealing schedule*"

função de custo, as "temperaturas" são calculadas através das seguintes fórmulas:

$$T_0 = f_{\min} + \frac{f_{\max} - f_{\min}}{10} \qquad \text{onde} \qquad \begin{cases} b = \dfrac{T_0 - T_f}{MT_0T_f} \\ M = \dfrac{50(n)(n-1)}{2} \end{cases}$$

$$T_f = f_{\min}$$

$$T_{i+1} = \frac{T_i}{1 + bT_i}$$

sendo n=número de ordens

- Experimentalmente, através do ajuste da "temperatura" inicial e a "temperatura final" de acordo com o desempenho do algoritmo [CHWIF (1994)].

- Função de decréscimo de "temperatura" – determina em que medida a probabilidade de aceitação se reduz; isto significa definir a forma da função de decréscimo da "temperatura", bem como todos os seus parâmetros.

A diretriz de determinação de uma função de decréscimo de "temperatura" deve contemplar o equilíbrio entre tempo de execução do algoritmo e qualidade da solução desejada. Funções de decréscimo rápido de "temperaturas" farão com que o SA se comporte de maneira semelhante a um algoritmo de otimização local enquanto funções mais lentas de decréscimo podem aumentar demasiadamente seu tempo de execução. Topologias não lineares da função podem desequilibrar a probabilidade de aceitação em distintos momentos do algoritmo, o que também pode deteriorar a qualidade da solução final.

As funções de "temperaturas" mais comuns são:

- Resfriamento proporcional – $T_i=\gamma T_{i-1}$ onde $(0<\gamma<1)$ é a taxa de resfriamento [KIRKPATRICK et al. (1983), POTTS; VAN WASSENHOVE (1991)];
- Resfriamento Suave – $T_i=T_i/(1+\eta T_{i-1})$, $\eta>0$, [VAN LAARHOVEN et al. (1992)];

- Resfriamento Linear – $T_i=((\lambda-i)/\lambda)\ T_{i-1}$; λ = constante (determinada experimentalmente [JOHNSON et al. (1989)];
- Resfriamento Logarítmico – $T_i=\lambda/(1+\log(i))$, λ = constante (número de reduções de "temperatura") [JOHNSON et al. (1989)];
- Resfriamento Exponencial – $T_i=\lambda\gamma^{i-1}$, onde λ e $\gamma<1$ são constantes.

iii. Critério de determinação do equilíbrio em cada estado

A cada temperatura, é necessário que se execute uma série de iterações para que o algoritmo convirja para uma solução "ótima" naquele determinado estado. O projeto do SA adaptado a um determinado problema deve prever quando se atinge este estado. Isto pode ser feito basicamente através das seguintes formas:

- a determinação de um número máximo de iterações a cada temperatura ("epoch length"). Este é a maneira mais comum adotada pelos pesquisadores. O "epoch" é proporcional ao tamanho da vizinhança analisada pelo problema na temperatura dada [JOHNSON et al. (1989), JOHNSON et al. (1991), VAN LAARHOVEN et al. (1992), MAMALIS; MALAGARDIS (1996), WILHELM; WARD (1987), OGBU; SMITH (1990)];
- implementações de critérios específicos ao problema [ZEGORDI et al. (1995)].

Contudo, há implementações como a de CONNOLY (1990) em que este equilíbrio não é testado e portanto há mudança de temperatura a cada iteração.

iv. Critério de Parada do algoritmo

A última decisão na implementação de um SA é quando se deve parar a execução do algoritmo. Diversas abordagens neste sentido podem ser utilizadas, contudo deve–se atentar para as iterações entre o critério de determinação do equilíbrio a cada temperatura e o critério de parada. A combinação dos dois critérios deve ser tal que se

evite o término antecipado do algoritmo, degradando a solução final ou que se prolongue muito sua execução com efeito mínimo sobre a qualidade da solução. Os critérios mais utilizados são:

- parar quando um determinado número de iterações de temperaturas foi atingido [MAMALIS; MALAGARDIS (1996)] – neste caso, define–se um número máximo de iterações ou perturbações na vizinhança com base no tamanho do problema (número de ordens) e quando este número é atingido, pára–se algoritmo;
- parar quando se atingir um determinado tempo de computação;
- parar quando se atinge uma temperatura final Tf, determinada empiricamente [MITTHENTHAL et al. (1996), VAN LAARHOVEN et al. (1992), POTS; VAN WASSENHOVE (1991), CHWIF (1994)]
- parar quando o número de tentativas exceder um valor máximo M [ZEGORDI et al. (1995)]
- critério de JOHNSON et al. (1989) – um contador é mantido e incrementado a cada vez que é terminada a execução de um "epoch" em uma determinada temperatura, e a porcentagem de movimentos aceitos nesta temperatura é menor ou igual a um parâmetro denominado MINPERCENT (percentual mínimo de movimentos aceitos para uma temperatura). Se uma solução melhor que a atual que está arquivada (a melhor até o momento) é encontrada, este contador zera. Se em algum momento da execução do algoritmo este contador atinge o valor 5, a execução do algoritmo é encerrada pois o sistema é considerado *congelado.*

Tendo escolhido os parâmetros acima, o algoritmo básico do "simulated annealing" consiste nos seguintes passos:

i. Gerar uma solução inicial S e computar seu custo;

ii. Selecionar uma "temperatura inicial" T_i dependente da probabilidade de aceitação inicial desejada;

iii. Inicializar o contador de epoch J=1 ;

iv. Enquanto J<M (máximo de iterações por "temperatura");

v. Gerar solução vizinha de S, S';

vi. Computar custo C(S´) e calcular Δ=C(S´)–C(S);

vii. Se Δ<0 então S=S´;

viii. Senão S=S´ com a probabilidade $\exp(-\Delta/T_k)$;

ix. .Se a condição de parada for satisfeita, parar. Caso contrário, reduzir a temperatura conforme evolução do resfriamento decidido ($T_k=rT_{k-1}$ para resfriamento constante), e ir para o passo iii.

As principais vantagens da aplicação do SA aos problemas de "scheduling" residem na relativa simplicidade do algoritmo, sua facilidade de adaptação às diversas instâncias do problema e a convergência assimtóptica ao ótimo sob determinadas condições [VAN LAARHOVEN et al. (1992)]. Entretanto, a implementação do SA é extremamente sensível à escolha de parâmetros, o que exige extensivo trabalho experimental para sua determinação [PARK; KIM (1998)]. Além disso, o SA costuma apresentar tempos de execução superiores aos outros algoritmos de busca na vizinhança e algumas heurísticas específicas de problemas para uma mesma performance. Em geral, as aplicações bem-sucedidas do SA aos problemas de "scheduling" apresentam as seguintes características:

- Encontram ótimos para problemas de dimensão reduzida e problemas de média complexidade;

- Para problemas maiores, encontram soluções comparáveis às melhores heurísticas;
- Encontram soluções melhores que algoritmos baseados em otimização local dentro do mesmo tempo de execução.

3.9.5 Heurísticas Baseadas em Gargalos

Os gargalos são as máquinas que apresentam ocupação total de sua capacidade ou carga (tempo de processamento necessário) num determinado horizonte de programação. As heurísticas baseadas em gargalos enfocam a identificação deste tipo de máquina (num problema n>=2) e procuram resolver iterativamente o problema relaxado para estas máquinas. Os principais métodos que utilizam este conceito são OPT [MORTON; PENTICO (1993)], "Shifiting Bottleneck" [ADAMS et al. (1988)] e "Bottleneck Dynamics" [MORTON; PENTICO (1993)].

3.9.5.1 OPT

OPT é a sigla de "Optimized Production Technology" (Tecnologia de Otimização da Produção) e se refere a um software criado por GOLDRATT (1989). Apesar de detalhes do software serem proprietários, é conhecido que seu processo envolve quatro etapas:

i. Determinação do recurso gargalo

Pode ser feito por 3 métodos:

- Através da máquina com maior utilização (maior soma dos tempos de processamento na fila);
- Através da resolução do problema de uma máquina para cada possível candidata;
- Através de um método heurístico rápido.

ii. Seqüenciar a máquina gargalo – significa resolver o problema por algum método exato ou heurística;

iii. Seqüenciar as máquinas anteriores ao gargalo – utilizando–se o mesmo método que no passo anterior, seqüenciam–se as máquinas anteriores ao gargalo (sob o ponto de vista do roteiro de produção) considerando a restrição de que as datas de finalização devem ser as mesmas datas de começo que no gargalo;

iv. Seqüenciar as máquinas restantes – utilizando–se as datas de finalização das atividades seqüenciadas nos passos ii e iii como restrição para as datas de início das máquinas restantes, resolve–se o problema de uma máquina para os demais recursos.

3.9.5.2 "Shifiting Bottleneck"

Este algoritmo, desenvolvido por ADAMS et al (1988), é aplicado ao problema de "makespan" de um "job-shop" e utiliza dois conceitos principais:

- a programação não–linear, com a resolução iterativa de um problema de várias variáveis através da fixação de uma única delas e sua otimização, relaxando–se as demais, e
- a definição de vários gargalos em ordem de importância, o que faz com que o processo de resolução iterativa (conceito acima) produza soluções locais melhores que a escolha arbitrária das variáveis fixas.

O algoritmo básico consiste nos seguintes passos:

i. resolver isoladamente o problema para cada uma das máquinas existentes no chão de fábrica;

ii. a máquina cujo valor de custo for mais alto será o gargalo primário;

iii. se a resolução do gargalo é ótima então o valor do custo é um limite inferior do problema,

iv. resolver o seguinte problema relaxado:

A. fixar a seqüência obtida no gargalo primário,

B. escolher uma máquina e relaxar as demais (aumentar infinitamente cada capacidade);

v. escolher outra máquina e executar o mesmo processo do passo iv até todas as máquinas tenham sido escolhidas,

vi. determinar a máquina com maior custo. Esta máquina será o gargalo secundário,

vii. executar o balanceamento dos gargalos através do processo:

A. fixar a seqüência do gargalo secundário e otimizar o primário,

B. fixar a seqüência do gargalo primário e otimizar o secundário,

C. repetir até não haver melhorias no objetivo;

viii. fixando os gargalos já encontrados, executar para outra máquina os passos iv a vi,

ix. balancear os gargalos já encontrados em processo semelhante ao passo vii,

x. repetir os passos viii e ix até todas as máquinas serem classificadas em níveis de gargalo,

xi. computar o resultado final. Se este resultado for melhor que o limite inferior encontrado, a solução é ótima.

Este procedimento foi denominado de "Shifiting Bottleneck I" (SB1) pelos autores. Eles desenvolveram um outro método (SB2) que utiliza enumeração parcial (tipo "beam–search") na busca de um gargalo. Cada máquina gargalo é um nó numa árvore e

dá origem a um número fixo de filhos que representam potenciais gargalos inferiores. A busca é controlada por uma função que penaliza a geração de nós cuja máquina desvia–se muito do gargalo primário.

Os autores utilizaram o algoritmo "branch and bound" desenvolvido por CARLIER; PINSON (1989) para o objetivo de atraso máximo, na resolução dos problemas relaxados de uma máquina.

Os resultados encontrados foram, em geral, superiores às heurísticas para os problemas resolvidos, sendo ainda encontrados valores ótimos para alguns problemas clássicos em tempo computacional competitivo em relação às heurísticas reportadas.

3.9.5.3 "Bottleneck Dynamics"

"Bottleneck Dynamics" é uma heurística que gera seqüências nas máquinas de acordo com prioridades geradas por uma análise de custo/benefício entre despachar a atividade e utilizar o recurso (máquina). O método requer para tal finalidade a estimação de custos das atividades e custos de utilização dos recursos (máquinas) e poder ser considerado um método de gargalo, pois suas regras de despacho tendem a priorizar as máquinas gargalo (que apresentam maior custo). Pode ser aplicado a problemas estáticos ou dinâmicos, de uma ou várias máquinas sendo mais apropriado a problemas dinâmicos de multi–máquinas. As suas resoluções podem abranger problemas de "scheduling", sequenciamento ou roteirização.

Os custos (preços) estimados para as atividades podem ser gerados pelos seguintes métodos:

- Um múltiplo fixo do tempo de processamento restante da ordem;
- Interação dos "lead-times" através de simulação;

- Cruzamento entre dados históricos ou de simulação às diversas condições de carregamento do chão de fábrica;
- Experiência humana.

Os custos (preços) dos recursos devem ser estimados no período ocupado de uma máquina (isto é, quando ela está carregada ou há fila). A idéia básica é calcular o custo adicional de cada atividade por unidade de tempo caso o recurso fique indisponível (devido, por exemplo, a quebras). Normalmente isto é conseguido através de simulações do chão de fábrica ou por busca no caso de problemas menores.

A partir dos preços estimados para as atividades e recursos e objetivos, pode–se derivar métodos para a geração de regras de prioridade para o "scheduling", expedição e roteamento. Alguns destes métodos, aplicações e respectivos resultados encontram–se detalhados em MORTON; PENTICO (1993).

4 MODELAGEM ADOTADA

4.1 Introdução

A metodologia adotada faz uma analogia entre os problemas de programação da produção e os problemas de "layout". Modela–se o problema de programação de produção como um problema de "layout", em que cada operação de cada ordem é um bloco que possui diferentes fluxos com outros blocos, fluxos estes que representam prioridades de posicionamento relativo entre as operações. Os blocos ou atividades possuem uma largura proporcional ao seu tempo de processamento e são dispostos linearmente em uma determinada ordenada, que indica à máquina onde serão processados. O arranjo resultante deste modelo pode ser interpretado como um gráfico Gantt, solução de um problema de "scheduling". A minimização dos momentos de transporte pode indiretamente medir o objetivo de mínimo "makespan" ou funcionar como uma heurística auxiliar em problemas de "scheduling" que possuem objetivos de data, como o objetivo de atraso ponderado.

4.2 O problema do planejamento do arranjo físico ou problema de "layout"

4.2.1 Definições

O problema de planejamento do arranjo físico ou "layout" consiste na determinação da alocação ótima de atividades para se conseguir uma produção ou execução de um serviço com menor custo de transporte. Por atividade pode–se entender, em um nível mais agregado de planejamento, a distribuição de fábricas e armazéns em uma região; em um nível mais operacional, a disposição de departamentos em uma planta e em um nível mais detalhado, a disposição de máquinas numa célula de produção.

Os problemas de arranjo físico podem ser classificados em:

- Novo arranjo - caracteriza–se pela inserção de novas atividades quando se tem ou não alguma atividade pré–alocada..
- Rearranjo - refere–se à redistribuição das atividades já existentes.
- Expansão/Redução - refere–se à mudança de características geométricas das atividades.

Várias metodologias e sistemas foram desenvolvidos para a resolução deste problema ao longo dos anos. Dentre essas metodologias, uma das mais conhecidas é a SLP ("Sistematic Layout Planning"), desenvolvida por MUTHER (1978). Esta metodologia utiliza duas ferramentas para analisar o "layout", através das perspectivas quantitativa e qualitativa, respectivamente:

- Cartas DE/PARA - estabelecem a medida de fluxo (interação entre duas atividades) através de uma matriz n x n, onde cada valor numérico corresponde à medida de interação (o fluxo de materiais, por exemplo) entre a atividade da coluna e a atividade da linha. Um exemplo de uma Carta DE/PARA pode ser encontrado na Figura 4.1
- Cartas de Interligações Preferenciais - estabelece mqualitativamente o grau de proximidade necessário entre as atividades. MUTHER (1978) estabeleceu uma convenção em que classifica em uma medida de prioridade denominada "closeness ratings", a proximidade relativa entre atividades. Esta classificação toma as seguintes denominações:
  - A ("Absolutely necessary") – se é absolutamente necessário que as atividades estejam próximas;
  - E ("Especially important") – se é especialmente importante que as atividades estejam próximas;

- I ("Important") – se é importante que as atividades estejam próximas;
- O ("Ordinary close") – se é pouco importante que as atividades estejam próximas;
- U ("Unimportant") – se é indiferente a proximidade das atividades
- X ("Indesejável") – se é indesejável a proximidade entre as atividades

PARA / DE	Atividade 1	Atividade 2	Atividade 3	Atividade 4
Atividade 1		10		20
Atividade 2			10	35
Atividade 3		5		10
Atividade 4			20	

Figura 4.1 Exemplo de Cartas DE / PARA

	Atividade 1	Atividade 2	Atividade 3	Atividade 4
Atividade 1		U	I	A
Atividade 2	U		U	O
Atividade 3	I	U		X
Atividade 4	A	O	X	

Figura 4.2 Exemplo de Cartas de Interligações Preferenciais

Nota–se que na Carta de Interligações Preferenciais somente há uma prioridade para cada par de atividades enquanto nas Cartas DE/PARA, os fluxos dependem do sentido do fluxo. Segundo MUTHER (1978), a primeira é mais apropriada para atividades não diretamente ligadas à produção (por exemplo, a área administrativa) enquanto que a última metodologia de análise é mais apropriada a atividades produtivas.

Os sistemas denominados CAL ("Computer Aided Layout") procuram através de heurísticas, métodos otimizantes ou de busca solucionar o problema. Em CHWIF

(1994) pode–se encontrar referências a vários destes sistemas bem como suas respectivas abordagens.

4.2.2 Formulação e Modelagem

A formulação de um problema de arranjo físico consiste na adequada escolha das seguintes características do problema:

4.2.2.1 Função Objetivo

Em geral, o objetivo de um problema de arranjo é modelado como uma função de custo da forma:

$$\min \; Z = \sum_{i=1}^{N} \sum_{t=1}^{N} f_{ij} \rho_{ij} a_{ij} + \sum_{t=1}^{N} v_t$$

onde:

f_{ij} é a medida de iteração (fluxo) entre as atividades i e j

a_{ij} é a medida de distância ou afastamento entre as atividades i e j

ρ_{ij} é o fator de conversão para custos entre as atividades i e j

v_t são os custos fixos de localização de cada atividade

Uma outra abordagem utilizada é o QAP ("Quadratic Assignment Problem") [WILHELM; WARD (1987)] cuja função pode ser escrita da seguinte forma:

$$Min\, Z = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} f_{ik} v_{ik} a_{jl} \mu_{ij} \mu_{kl} + \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} \mu_{ij}$$

$$s.a. \quad \sum_{i=1}^{n} \mu_{ij} = 1$$

$$\sum_{j=1}^{n} \mu_{ij} = 1$$

$$\mu_{ij} \in \{0,1\}$$

onde

b_{ij} – custo fixo de posicionar atividade i na localização j

v_{ik} – custo unitário da medida de iteração entre as atividades i e k

f_{ik} – medida de iteração entre as atividades i e k (fluxo de materiais)

a_{jl} – distância ou afastamento entre as localizações j e l

μ_{ij}= 1 se atividade i for posicionada na posição j

0 caso contrário

A principal desvantagem desta última formulação em relação à primeira reside no fato de o QAP se aplicar à modelagem discreta de atividades de mesma área, uma limitação muito restritiva na prática.

4.2.2.2 Modelagem geométrica das atividades

As principais abordagens de modelagem geométrica entre as atividades são:

- Pontual: cada atividade - é um ponto no plano,
- Quadrado - cada atividade possui a mesma área e mesma orientação,
- Retângulos - a abordagem de retângulos mais flexível é a de TAM; LI (1992) onde cada atividade retangular pode ser posicionada horizontalmente ou verticalmente,
- Outros: formato em L, polígonos.

No presente trabalho, adota–se a modelagem geométrica por retângulos.

4.2.2.3 Modelagem geométrica do contorno

As atividades devem ser localizadas em área que possuam ou não contornos. No último caso o problema é não restrito, e as atividades podem ser livremente localizadas em qualquer ponto do espaço. No primeiro caso, o contorno é normalmente modelado como um retângulo ou polígono. Quanto maior a complexidade do contorno, mais difícil ficará sua modelagem e resolução, porém maior será a aderência do problema à

realidade. Adota–se a modelagem geométrica de retângulo para o contorno. Tal abordagem justifica–se pelo fato de que o contorno representa apenas uma restrição temporal (posicionamento na abscissa X), no qual as atividades não poderão ser posicionadas anteriormente à borda esquerda, em tempos negativos (X<0), nem tampouco em horizontes de tempo superiores ao período de programação estabelecido (por exemplo, uma semana), cuja restrição é estabelecida pela borda direita do contorno.

4.2.2.4 Medida de distância

Na literatura, são utilizadas basicamente duas medidas de distância:

- Distância ou afastamento retangular ou de "Manhattan" - sendo i e j as atividades e x e y as coordenadas de seus baricentros, temos:

$$a_{ij} = |x_i - x_j| + |y_i - y_j|$$

- Distância ou afastamento euclidiano - pode ser descrita por

$$a_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Adota–se neste caso o conceito de distância retangular, pois somente é relevante para a modelagem a distância linear entre as atividades na abscissa X.

4.2.2.5 Método e Técnica de Resolução

Para as formulações descritas acima, aplicam–se métodos de solução que podem ser classificados em:

- Métodos Otimizantes - dentre os métodos otimizantes o mais popular é sem dúvida o "branch & bound", já explicado anteriormente. A vantagem deste método é a garantia de atingir–se o ótimo. Contudo, problemas de maior complexidade

(número de atividades>15) podem levar a um excessivo tempo de execução devido à característica NP destes problemas, segundo GILMORE (1962).

- Métodos Heurísticos - para contornar o problema da complexidade computacional, inúmeros métodos heurísticos foram desenvolvidos para o problema de layout. Um dos mais conhecidos é o "steepest descendent pairwise interchange", desenvolvido por ARMOUR; BUFFA (1963) e utilizado no sistema CRAFT. Consiste em efetuar trocas entre pares de atividades (análogo ao "pairwise interchange" anteriormente explicado) para uma configuração e escolher aquela troca que gere maior redução no custo. Este processo repete–se até que não seja mais possível reduzir os custos.
- Métodos Hierárquicos - consistem em dividir por algum critério o problema de layout em subproblemas menores e resolvê–los por um método otimizante (como um "branch and bound"). Este conceito é utilizado no trabalho de TAM; LI (1991). Porém, como a otimização das partes não garante a otimização do todo, esta metodologia não garante a convergência ao ótimo.
- Métodos de Busca na Vizinhança ("Neighbourhood Search") - coincidem com as explicações de métodos gerais de busca estendida como o "Beam Search" (BS), "Tabu Search" (TS), "Genetic Algorithms" (GA) e "Simulated Annealing" (SA) mais extensamente discutidos anteriormente para o problema de "scheduling". Dentre estes métodos, este presente trabalho irá enfocar no SA que já apresentou resultados satisfatórios para o problema [JAJODIA et al. (1991), KOUVELIS; CHIANG (1992) e CHWIF (1994)].

A aplicação dos métodos ao problema do "layout" bem como suas respectivas formulações e modelagens, são extensivamente discutidos por CHWIF (1994).

4.3 A analogia entre o "layout" e o "scheduling"

4.3.1 A analogia e sua formulação

O problema de "scheduling", descrito na seção 3.3, pode ser modelado da seguinte forma:

Supondo que cada ordem de produção ou serviço i seja composta por um conjunto de operações j, cada uma podendo ser executada em uma única máquina k, denominamos cada operação ou atividade como O_{ijk}. Cada O_{ijk} pode ser modelada como uma atividade (bloco) cujo baricentro é representado pelas coordenadas x_{ijk} e y_{ijk} a ser disposta num problema de "layout". Este bloco é na verdade um retângulo de altura h_{ijk} constante e largura w_{ijk} proporcional ao seu tempo de processamento. Sua disposição num plano (planta) deve ser tal que sua ordenada do baricentro (y_{ijk}) determine a máquina onde a ordem vai ser processada e a abscissa (x_{ijk}) defasada de metade do tempo de operação ($w_{ijk}/2$) determine o tempo de sua entrada em processamento. É importante esclarecer que o presente trabalho não procura resolver problemas de alocação quadrática (QAP), ou seja, o problema estabelece uma relação unívoca entre a ordenada da atividade O_{ijk} (bloco) e a máquina k, de modo que um bloco somente pode ocupar uma ordenada fixa (y_{ij}), esta última determinada pelo roteiro de produção (isto é, a máquina(k) onde a atividade O_{ijk} vai ser processada).Uma vez definida esta restrição, os blocos podem ocupar qualquer posição na abscissa (x), mantendo–se a ordenada (y) fixa. Neste caso podemos desconsiderar o índice k, visto que o par (i, j) por si define a máquina k (através ordenada y_{ij}) na qual a operação pode ser processada. Uma ilustração da modelagem apresentada pode ser melhor visualizada na figura 4.3:

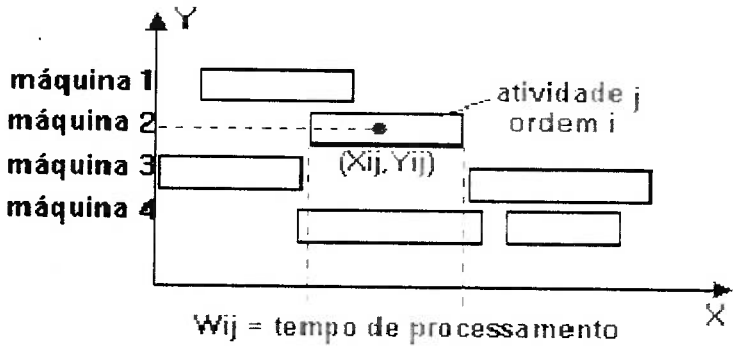

Figura 4.3 Analogia entre "layout" e "scheduling"

A medida de interação entre duas atividades é o produto entre o fluxo de materiais entre duas atividades e a distância entre elas. Na analogia com o "scheduling", a determinação dos valores dos fluxos entre as atividades (operações) e da medida de distância fará com que sejam respeitados os roteiros de produção bem como a produção de soluções exeqüíveis. Para tanto as seguintes premissas são adotadas:

- Fluxo de Materiais entre Ordens - seja f_{ijpq} o fluxo de materiais hipotético entre os blocos (atividades) O_{ijk} e O_{pqr} dispostos no plano de forma semelhante à Figura 4.3. Se duas operações O_{ijk} e O_{pqr} pertencem à mesma ordem (isto é, i = p), então o fluxo entre elas apresentará um valor alto. Se duas operações O_{ijk} e O_{pqr} não pertencem à mesma ordem mas pertencem à mesma máquina (i <> p; k = r), então o fluxo entre elas apresentará um valor baixo, e, caso as operações não possuam máquinas ou ordens em comum, o fluxo entre elas será nulo. Esta abordagem é justificada, pois num problema de "layout", as atividades que possuem maior fluxo entre elas tenderão a ficar mais próximas entre si.
- Medida de Distâncias - utiliza–se o conceito de distância ou afastamento de "Manhattan", preocupando–se apenas com o afastamento horizontal entre as ordens, já que a abordagem não prevê roteiros alternativos (ou seja mudanças de máquinas) e, portanto, as ordens sempre apresentarão uma ordenada fixa.

I. Operações não pertencentes à mesma ordem mas pertencentes à mesma máquina: Sejam O_{ijk} e O_{pqr}, duas operações tal que I <> p e k = r. Neste caso a distância ou afastamento (a_{ijpq}) são calculados por:

$$a_{ijpq} = \min\left(\left|(x_{ijk} + \frac{w_{ijk}}{2}) - (x_{pqk} - \frac{w_{pqk}}{2})\right|, \left|(x_{ijk} - \frac{w_{ijk}}{2}) - (x_{pqk} + \frac{w_{pqk}}{2})\right|\right)$$

Esta fórmula deriva da observação de que para uma determinada máquina, o menor valor entre a distância do extremo mais a direita de uma operação e o mais à esquerda da outra e vice versa, mede na realidade a inserção de espera numa determinada máquina, o que, para o objetivo de "makespan" deveria ser minimizado. Tal observação pode ser visualizada na Figura 4.4.

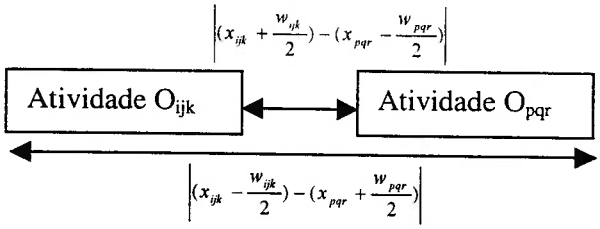

Figura 4.4 Cálculo de distância para operações de ordens distintas e mesma máquina

II. Operações de ordens distintas pertencentes a máquinas distintas - para o caso de operações de ordens distintas e máquinas distintas, o cálculo da distância é o mesmo que para operações de ordens distinta e mesma máquina e não influencia no comportamento do algoritmo visto que o fluxo entre operações de ordens distintas e máquinas diferentes é nulo (f_{ijkpqr}=0 para i<>p e k<>r).

- Função Objetivo - para a modelagem dos fluxos (f_{ijpq}) e das distâncias (a_{ijpq}) explicados acima, adota–se a seguinte função objetivo genérica (Z), adaptada da seção 4.2.2.

Z = A + B + C + D, onde:

- "A" – corresponde à parcela de minimização dos custos de transporte, ou seja, a soma do produto dos fluxos com as respectivas distâncias para todas as operações. De acordo com a analogia já descrita, esta parcela fará com que as atividades (ordens) que possuam uma alta iteração entre si tendam a permanecer próximas, o que significa que a seqüência gerada tenderá a reduzir os tempos ociosos entre as operações. A parcela "A" pode ser calculada por:

$$A = \alpha_1 \cdot \sum_{i=1}^{n} \sum_{j=1}^{t(i)} \sum_{p=1}^{n} \sum_{q=1}^{t(p)} f_{ijpq} \cdot a_{ijpq}$$

sendo que:

f_{ijpq} é o fluxo entre as atividades O_{ijk}, O_{pqr}

a_{ijpq} é a distância ou afastamento medidos entre as atividades O_{ijk} e O_{pqr} (definidos na seção anterior)

α_1 é o peso do custos de transporte da função objetivo (Z)

n é o número de ordens

t(i) é o número total de operações de cada ordem i.

- "B" – a parcela "B" refere–se à função de penalização necessária pois o problema de "layout" em cuja analogia se baseia esta abordagem resolve problemas sem restrição, o que quer dizer que o algoritmo pode produzir soluções não plausíveis ("unfeasible"). Estas soluções "unfeasible" são quaisquer soluções em que haja sobreposição física de duas atividades ou blocos, que se traduz no "scheduling" como o processamento de duas atividades na mesma máquina ao mesmo tempo.

Um exemplo de solução "unfeasible" é ilustrado na Figura 4.5. A parcela "B" é calculada por:

$$B = \alpha_2 . \sum_{i=1}^{n} \sum_{j=1}^{t(i)} \sum_{p=1, i \neq p}^{n} \sum_{q=1, q \neq j}^{t(p)} I_{ijpq}$$

$$I_{ijpq} = \begin{cases} E_{ijpq} & E_{ijpq} > 0 \quad \text{onde } E_{ijpq} = \frac{w_{ij}+w_{pq}}{2} - \left| x_{ij} - x_{pq} \right| \\ 0 & \text{caso contrário} \end{cases}$$

sendo que:

w_{ij} é a largura da atividade O_{ijk}, que significa o tempo de processamento no problema de "scheduling"

h_{ij} é a altura da atividade O_{ijk}, mantida constante para todas as atividades

x_{ij},y_{ij} são as coordenadas do baricentro do bloco O_{ijk}

α_2 é o peso da função de penalidade

n é o número de ordens

t(i) é o número total de operações de cada ordem i

Nota–se que I_{ijpq} é maior que zero se as operações de mesma máquina se sobrepuserem pois a parcela E_{ijpq}, que é diferença entre a soma da média das larguras de duas atividades e o módulo das abcissas do baricentro, torna–se positiva. Tal fato é melhor ilustrado na figura 4.5:

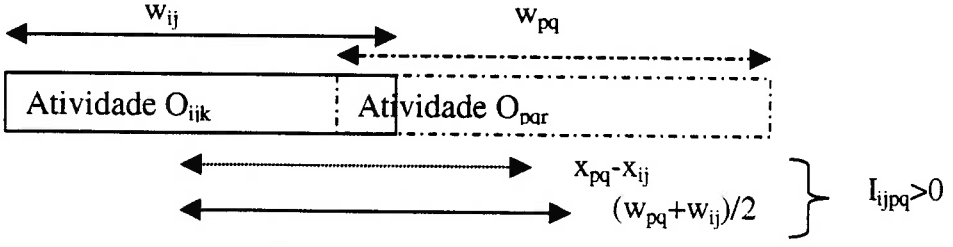

Figura 4.5 Exemplo de sobreposição de atividades

- "C" – Refere–se à penalidade de seqüência, para problemas do tipo "flow–shop" ou "job–shop". Tal penalidade é necessária pois a modelagem resolve problemas sem restrição e, portanto, é preciso penalizar soluções que desrespeitem a seqüência tecnológica das operações dentro da ordem. A parcela "C" é calculada por:

$$C = \alpha_3 . \sum_{i=1}^{n} \sum_{j=1}^{t(i)} \sum_{l=1; l \neq j}^{t(i)} \theta_{ijl}$$

$$\theta_{ijl} = \begin{cases} 0 & \text{se } l > j \text{ e } (x_{il} - w_{il}/2) > (x_{ij} + w_{ij}/2) \text{ ou se } j > l \text{ e } (x_{ij} - w_{ij}/2) > (x_{il} + w_{il}/2) \\ 1 & \text{caso contrário} \end{cases}$$

sendo que:

w_{ij} é a largura da atividade O_{ijk}, que significa o tempo de processamento no problema de "scheduling"

x_{ij},y_{ij} são as coordenadas do baricentro do bloco O_{ijk}

α_3 é o peso da função de penalidade da seqüência (para problemas de multi–máquinas)

n é o número de ordens

t(i) é o número total de operações de cada ordem i

θ_{ijl} – é a função que toma valor 1 se dentro da ordem i, as operações j e l desrespeitam a seqüência tecnológica, o que ocorre se na seqüência da ordem i, a operação j antecede l mas l está posicionada antes de j.

- "D" – a parcela "D" refere–se ao objetivo específico de "scheduling" e pode tomar quaisquer das formulações descritas na seção 3.5. Especificamente para este trabalho, foram utilizadas três funções distintas para o problema de "scheduling":

  - "Makespan" (C_{max}) - a parcela "D" fica modelada como:

$$D = \alpha_4 (\underset{\forall i,j}{Max}(x_{ij}) - \underset{\forall i,j}{Min}(x_{ij}))$$

sendo que:

$Max(x_{ij})$ é o valor máximo da coordenada x que pode tomar qualquer atividade (i,j)

$Min(x_{ij})$ é o valor mínimo da coordenada x que pode tomar qualquer atividade (i,j).

- Atraso total (T_t):

Seja a data de finalização de uma ordem ("due date") representada por d_i.

$$D = \alpha_4 . \sum_{i=1}^{n} \max\left(\sum_{j=t(i)}^{t(i)} \left(x_{ij} + \frac{w_{ij}}{2} - d_i \right); 0 \right)$$

sendo que:

w_{ij} é a largura da atividade O_{ijk}, que significa o tempo de processamento no problema de "scheduling"

x_{ij},y_{ij} são as coordenadas do baricentro do bloco O_{ijk}

α_4 é o peso da função objetivo de "scheduling" ("makespan" ou atraso ponderado);

n é o número de ordens

t(i) é o número total de operações de cada ordem i

d_i é a data de finalização ("due date") da ordem i.

- Atraso ponderado (T_{wt}):

Seja a data de finalização de uma ordem ("due date") representada por d_i e seu respectivo peso ou prioridade por U_i

$$D = \alpha_4 . \sum_{i=1}^{n} \max\left(\sum_{j=t(i)}^{t(i)} \left(\left(x_{ij} + \frac{w_{ij}}{2} - d_i \right) U_i \right); 0 \right)$$

sendo que:

w_{ij} é a largura da atividade O_{ijk}, que significa o tempo de processamento no problema de "scheduling"

x_{ij},y_{ij} são as coordenadas do baricentro do bloco O_{ijk}

α_4 é o peso da função objetivo de "scheduling" ("makespan" ou atraso ponderado)

n é o número de ordens

t(i) é o número total de operações de cada ordem i

d_i é a data de finalização ("due date") da ordem i

U_i é o valor do peso ou prioridade de cada ordem i

Dentro desta forma genérica, a função objetivo Z pode tomar formas diferentes de acordo com o problema de "scheduling" abordado. Particularmente dentro do escopo deste trabalho, foram adotadas três funções objetivos para três classes de problemas de "scheduling" distintos, em termos de complexidade ("job-shop", "flow-shop" e única máquina). Tais problemas e suas respectivas funções de custo são:

- "Makespan" (C_{max}) para problemas de multi–máquina dos tipos "flow–shop" e "job-shop"

Neste caso, a função objetivo possui os quatro componentes explicados anteriormente, tomando a seguinte forma:

$$Z = \alpha_1 . \sum_{i=1}^{n} \sum_{j=1}^{t(i)} \sum_{p=1}^{n} \sum_{q=1}^{t(p)} f_{ijpq} . a_{ijpq} + \alpha_2 . \sum_{i=1}^{n} \sum_{j=1}^{t(i)} \sum_{p=1, i \neq p}^{n} \sum_{q=1, q \neq j}^{t(p)} I_{ijpq} + \alpha_3 . \sum_{i=1}^{n} \sum_{j=1}^{t(i)} \sum_{l=1; l \neq j}^{t(i)} \theta_{ijl} + \alpha_4 \left(\underset{\forall i,j}{Max}(x_{ij}) - \underset{\forall i,j}{Min}(x_{ij}) \right)$$

$$I_{ijpq} = \begin{cases} E_{ijpq} & \text{E}_{\text{ijpq}} > 0 \quad \text{onde } E_{ijpq} = \frac{w_{ij} + w_{pq}}{2} - \left| x_{ij} - x_{pq} \right| \\ 0 & \text{casocontrário} \end{cases}$$

$$\theta_{ijl} = \begin{cases} 1 & \text{se } j < l \text{ e } (x_{ij} - w_{ij}/2) > (x_{il} + w_{il}/2) \\ 0 & \text{casocontrário} \end{cases}$$

- Atraso ponderado(T_{wt}) para problemas de única máquina

Neste caso, tornam–se desnecessárias as parcelas A e C pelo fato de tratar–se de um problema com uma só máquina. Do mesmo modo, a parcela D tem que modelar o objetivo T_{wt}, e assim a função objetivo Z fica da seguinte forma:

$$Z=\alpha_2.\sum_{i=1}^{n}\sum_{j=1}^{t(i)}\sum_{p=1,i\neq p}^{n}\sum_{q=1,q\neq j}^{t(i)}I_{ijpq}+\alpha_4.\sum_{i=1}^{n}\max\left(\sum_{j=t(i)}^{t(i)}\left(\left(x_{ij}+\frac{w_{ij}}{2}-d_i\right)U_i\right);0\right)$$

$$I_{ijpq}=\begin{cases}E_{ijpq} & \mathrm{E_{ijpq}}>0 \quad \text{onde} \quad E_{ijpq}=\frac{w_{ij}+w_{pq}}{2}-\left|x_{ij}-x_{pq}\right| \\ 0 & \text{caso contrário}\end{cases}$$

➢ Atraso total(T_t) para problemas flow shop

Neste caso, torna–se desnecessária a parcela A, pelo fato de tratar–se de um problema com datas. Do mesmo modo, a parcela D tem que modelar o objetivo T_t. Assim, a função objetivo Z fica da seguinte forma:

$$Z=\alpha_2.\sum_{i=1}^{n}\sum_{j=1}^{t(i)}\sum_{p=1,i\neq p}^{n}\sum_{q=1,q\neq j}^{t(p)}I_{ijpq}+\alpha_3.\sum_{i=1}^{n}\sum_{j=1}^{t(i)}\sum_{l=1;l\neq j}^{t(p)}\theta_{ijl}+\alpha_4.\sum_{i=1}^{n}\max\left(\sum_{j=t(i)}^{t(i)}\left(x_{ij}+\frac{w_{ij}}{2}-d_i\right);0\right)$$

$$I_{ijpq}=\begin{cases}E_{ijpq} & \mathrm{E_{ijpq}}>0 \quad \text{onde} E_{ijpq}=\frac{w_{ij}+w_{pq}}{2}-\left|x_{ij}-x_{pq}\right| \\ 0 & \text{caso contrário}\end{cases}$$

$$\theta_{ijl}=\begin{cases}1 & \text{se } j<l \text{ e} (x_{ij}-w_{ij}/2)>(x_{il}+w_{il}/2) \\ 0 & \text{caso contrário}\end{cases}$$

4.3.2 Modelo de Otimização Adotado

O modelo de otimização adotado foi o "Simulated Annealing" (SA), já discutido na seção 3.9.4. O uso de "Simulated Annealing" justifica–se pelas seguintes características:

- É um algoritmo simples de modelar e codificar;
- Possui extensiva aplicação a problemas de natureza combinatória dentro da pesquisa operacional [KOULAMAS et al. (1994)];

- Converge para o ótimo de acordo com apropriada escolha de parâmetros [VAN LAARHOVEN et al (1992)].

Os parâmetros a serem determinados empiricamente na abordagem do SA utilizada são:

- Temperaturas Inicial e Final – determinadas pela probabilidade de aceitação de soluções que se deseja no começo e no final da execução do algoritmo. A probabilidade de aceitação em cada temperatura pode ser calculada por PA= 1 – (R/E) onde PA – probabilidade de aceitação, R – número de rejeições, E – tamanho do "epoch lenght".
- Relação de Decréscimo da Temperatura – utiliza–se a abordagem de decréscimo proporcional da temperatura ($T_i=rT_{i-1}$), sendo r determinado experimentalmente.
- Tamanho do "Epoch" – o tamanho do "epoch" (número de iterações por temperatura) é determinado como um múltiplo M do tamanho do problema (número total de operações) E = M x n, onde E= número de iterações por temperatura, n = número de operações do problema, M=constante determinada experimentalmente.

O algoritmo pode ser iniciado com uma solução escolhida aleatoriamente ou através de uma heurística. Dentro das escolhas específicas do algoritmo, foram introduzidos dois mecanismos de geração de vizinhanças que funcionam alternadamente:

- O primeiro é análogo ao conhecido método de geração "pairwise interchange" [ARMOUR; BUFFA (1963)], que consiste, neste caso, em escolher aleatoriamente duas atividades e trocá–las de posição (em relação aos respectivos baricentros).
- O segundo método consiste em escolher arbitrariamente uma ordenada (máquina) e uma atividade(operação) pertencente a ela e a "adiantar" ou "atrasar" por um montante Δ, sendo Δ o "passo de movimentação" da atividade no tempo. Tal mecanismo de geração de vizinhanças foi denominado "linear move".

O procedimento de determinação da vizinhança do algoritmo, denominado LEO ("Linear Exchange Optimization"), funciona da seguinte forma:

i. Inicializar solução,

ii. Se vizinhança do passo anterior for "pairwise exchange", gerar vizinhança por "linear move",

iii. Se vizinhança do passo anterior for "linear move", gerar vizinhança por "pairwise exchange",

iv. Senão escolher aleatoriamente entre os dois mecanismos

5 ESTUDOS DE CASOS

Os resultados indicados nos testes detalhados a seguir foram produto da melhor saída de um sistema de simulação implementado no Microsoft Visual Basic 6.0. Os exemplos foram executados em computador Pentium III 600 Mhz. Os tempos de execução variaram com o tamanho do problema. Para problemas menores (10 atividades), o algoritmo executa em poucos minutos. O tempo de execução é descrito em cada teste.

A metodologia experimental adotada consistiu nos seguintes passos:

Passo 1: Ajuste de Parâmetros

Determinação parâmetros apropriados ao algoritmo de modo a obter soluções exeqüíveis ("feasible"). O procedimento adotado foi:

i. Ajuste inicial de parâmetros - tendo por base experimentos conduzidos anteriormente sobre o SA, como o de JOHNSON et al. (1989), e também através execuções teste inciais do algoritmo, partem–se de valores da seguinte ordem:

- Probabilidade de aceitação inicial do SA: 60%. Através deste nível de aceitação inicial pode–se estimar a temperatura inicial, segundo métodos discutidos na seção 3.9.4;
- α_1=1
- α_2=1
- α_3=10.000 (um valor alto comparativamente a α_1, α_2, α_4)
- α_4=1
- Temperatura final do SA= 1
- Passo de Movimentação (Δ)= 1
- "Epoch Length"= 20

ii. Execução do algoritmo

iii. Análise da solução encontrada sob os seguintes aspectos:

- Se a solução final tem custos de penalidade de sobreposição, aumenta–se α_2;
- Se a solução final tem custos de penalidade de seqüência, aumenta–se α_3;
- Se a solução produz resultados distantes de um limite inferior calculado (da ordem de 20% piores), limite este determinado por uma heurística (como as regras "SWPT" e "EDD") ou por resultados publicados anteriormente, ajusta–se para cima os coeficientes α_1 e α_4 (que se referem ao objetivo da função) e os parâmetros de epoch length, temperaturas iniciais e finais de modo a obter uma "annealing" mais lento;
- Se na temperatura final, a probabilidade de aceitação é superior a 5%, diminui–se a temperatura final;
- Se a temperatura inicial produz probabilidades de aceitação superiores a 80%, diminui–se a temperatura inicial;
- Se o algoritmo produz soluções com muita inserção de espera, ou seja, soluções cujas atividades consecutivas dentro da mesma máquina se encontram separadas, diminui–se o passo de movimentação;

iv. Atualizam–se os parâmetros caso haja modificações segundo regras acima e retorna ao item ii, senão termina.

Para problemas de maior complexidade foram necessários, de maneira geral, de 5 a 10 experimentos para encontrar–se empiricamente parâmetros que produzissem resultados consistentes, ou seja, soluções finais exeqüíveis.

Passo 2: Execução do Algoritmo

Com os parâmetros determinados no passo anterior, cada problema foi executado 10 vezes. Os resultados mostrados a seguir são o melhor resultado das 10 execuções, o que se justifica pela característica aleatória do algoritmo. Os outros resultados são gravados para efeitos de comparação da variabilidade da solução e convergência do algoritmo.

5.1 Resultados para problemas "job–shop" e "flow–shop" com objetivo de mínimo "makespan"

A. Problema 1

Este primeiro problema simples foi proposto para testar a validade do método proposto. É composto de 3 ordens, 6 atividades e 3 máquinas, como descrito na tabela 5.1.

ID	ORDEM (i)	OPERAÇÃO (j)	MÁQUINA (k)	NOTAÇÃO	P_{ijk}
J1	1	1	1	O111	30
J2	1	2	2	O122	20
J3	1	3	3	O133	10
J4	2	1	2	O212	25
J5	2	2	3	O223	15
J6	3	1	3	O313	10

Tabela 5.1: Dados do problema 1

Os parâmetros ajustados para este problema foram:

PARÂMETRO	VALOR
Peso da Função Custo ($\alpha 1$)	1
Peso da Penalidade ($\alpha 2$)	5
"Epoch Length" (M)	10
Temperatura Inicial	100
Temperatura Final	0,1
Constante de Penalidade da Seqüência ($\alpha 3$)	10000
Constante do "Makespan" ($\alpha 4$)	40
Passo de Movimentação (Δ)	1
Decremento da Temperatura(r)	0,95

Tabela 5.2 : Parâmetros do problema 1

Além disso, adotaram–se os seguintes valores de fluxo na etapa de ajustes dos parâmetros:

- f_{ijpq}=1 entre atividades de mesma máquina
- f_{ijpq}=10 entre atividades da mesma ordem
- f_{ijpq}= 0 para outras atividades

O resultado obtido como um Gráfico de Gantt pode ser visto na figura 5.1 e é detalhado na Tabela 5.3. O tempo de fluxo total ("makespan") resultante é 60 unidades de tempo e o custo total é de 3.434,99. O algoritmo executa em menos de 1 minuto. A solução encontrada é ótima, o que pode ser determinado por enumeração total.

ID	ORDEM (i)	OP (j)	MÁQUINA (k)	NOTAÇÃO	P_{ijk}	S_{ijk}	D_{ijk}
J1	1	1	1	O111	30	0	30
J2	1	2	2	O122	20	30	50
J3	**1**	**3**	**3**	**O133**	**10**	**50**	**60**
J4	2	1	2	O212	25	5	30
J5	2	2	3	O223	15	33	48
J6	3	1	3	O313	10	21	31

Tabela 5.3: Solução detalhada do problema 1

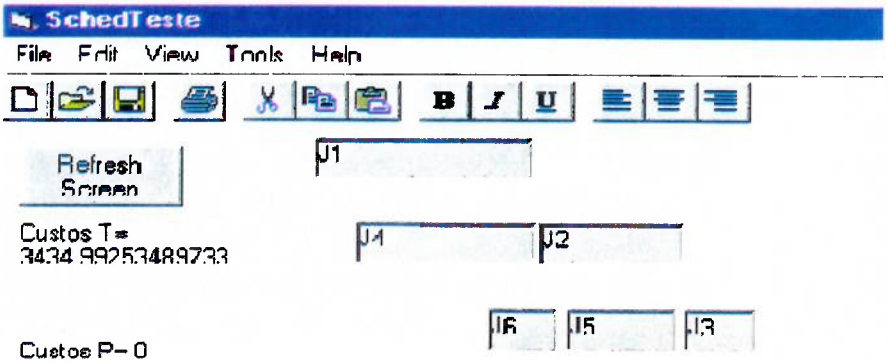

Figura 5.1 Gantt resultante do problema 1

B. Problema 2.

Este problema foi montado com 4 ordens e 10 atividades alocadas a 4 máquinas distintas. De acordo com a notação usada anteriormente, a representação deste problema pode ser vista na Tabela 5.4:

Id	ORDEM (I)	OPERAÇÃO (j)	MÁQUINA (k)	NOTAÇÃO	P_{ijk}
J1	1	1	4	O_{114}	30
J2	1	2	3	O_{123}	20
J3	1	3	2	O_{132}	40
J4	2	1	1	O_{211}	10
J5	2	2	2	O_{222}	35
J6	3	1	1	O_{311}	15
J7	3	2	2	O_{322}	20
J8	3	3	3	O_{333}	25
J9	4	1	4	O_{414}	15
J10	4	2	3	O_{423}	20

Tabela 5.4: Dados do problema 2

Os parâmetros utilizados para este problema foram:

PARÂMETRO	VALOR
Peso da Função Custo ($\alpha 1$)	1
Peso da Penalidade ($\alpha 2$)	20
"Epoch Length"	12
Temperatura Inicial	8000
Temperatura Final	5
Constante de Penalidade da Seqüência ($\alpha 3$)	10000
Constante do "Makespan" ($\alpha 4$)	40
Passo de Movimentação (Δ)	1
Decremento da Temperatura	0,95

Tabela 5.5 : Parâmetros do problema 2

O seguinte procedimento de determinação dos fluxos foi adotado após exeperimentos iniciais com o problema:

- f_{ijpq}=10 entre atividades de mesma máquina
- f_{ijpq}= 0 para outras atividades

O gráfico Gantt resultante pode ser visualizado na figura 5.2 e encontra–se detalhado na Tabela 5.6.

Id	ORDEM (i)	OP (j)	MÁQUINA (k)	NOTAÇÃO	P_{ijk}	S_{ijk}	D_{ijk}
J1	1	1	1	O_{111}	30	5	35
J2	1	2	2	O_{122}	20	38	58
J3	**1**	**3**	**3**	O_{133}	**40**	**65**	**105**
J4	2	1	4	O_{214}	10	0	10
J5	2	2	3	O_{223}	35	10	45
J6	3	1	4	O_{314}	15	16,5	31,5
J7	3	2	3	O_{323}	20	45	65
J8	3	3	2	O_{332}	25	78,5	103,5
J9	4	1	1	O_{411}	15	35,5	50,5
J10	4	2	2	O_{422}	20	58	78

Tabela 5.6: Solução detalhada do problema 2

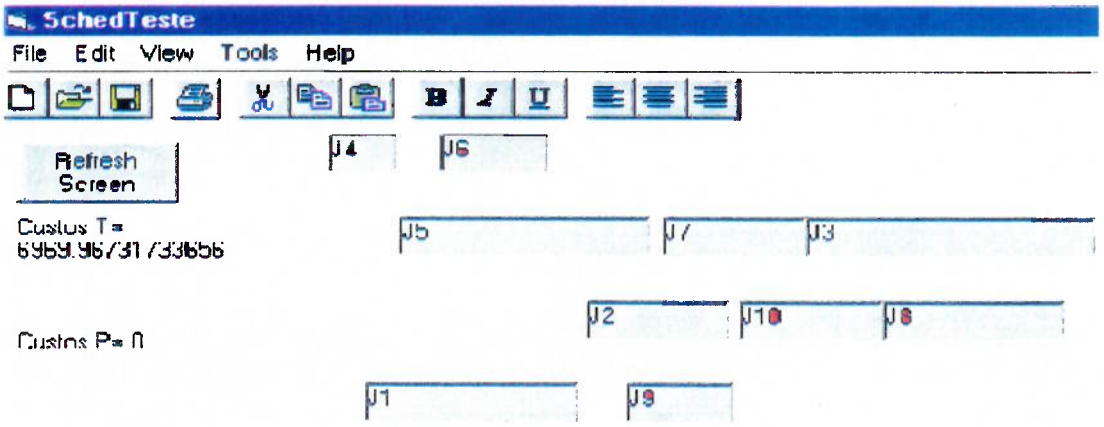

Figura 5.2 Gantt resultante do problema 2

O resultado obtido minimiza o tempo máximo de fluxo (que pode ser obtido por enumeração total). Nota–se que as operações 4,5,7 e 3 formam o caminho crítico do problema e portanto determinam o "makespan", que neste caso é de 105 unidades de tempo. A solução, obtida através de semente inicial aleatória, apresenta um custo total de 6.969,36. O algoritmo executa em cerca de 4 minutos.

C. Problema 3

O problema 3 é um problema encontrado no trabalho de FISCHER; THOMPSON (1963). Este problema consiste em um "flow–shop" de 6 ordens com 6 operações cada ordem e 6 máquinas, totalizando 36 operações ou atividades, descrito na Tabela 5.7:

Id	ORDEM (i)	OPERAÇÃO (j)	MÁQUINA (k)	NOTAÇÃO	P_{ijk}
J1	1	1	3	O_{113}	1
J2	1	1	1	O_{121}	3
J3	1	1	2	O_{132}	6
J4	1	1	4	O_{144}	7
J5	1	1	6	O_{156}	3
J6	1	1	5	O_{165}	6
J7	2	2	2	O_{212}	8
J8	2	2	3	O_{223}	5
J9	2	2	5	O_{235}	10
J10	2	2	6	O_{246}	10
J11	2	2	1	O_{251}	10
J12	2	2	4	O_{264}	4
J13	3	3	3	O_{313}	5
J14	3	3	4	O_{324}	4
J15	3	3	6	O_{336}	8
J16	3	3	1	O_{341}	9
J17	3	3	2	O_{352}	1
J18	3	3	5	O_{365}	7
J19	4	4	2	O_{412}	5
J20	4	4	1	O_{421}	5
J21	4	4	3	O_{433}	5
J22	4	4	4	O_{444}	3
J23	4	4	5	O_{455}	8
J24	4	4	6	O_{466}	9
J25	5	5	3	O_{513}	9
J26	5	5	2	O_{522}	3
J27	5	5	5	O_{535}	5
J28	5	5	6	O_{546}	4
J29	5	5	1	O_{551}	3
J30	5	5	4	O_{564}	1
J31	6	6	2	O_{612}	3
J32	6	6	4	O_{624}	3
J33	6	6	6	O_{636}	9
J34	6	6	1	O_{641}	10
J35	6	6	5	O_{655}	4
J36	6	6	3	O_{663}	1

Tabela 5.7: Dados do problema 3

Os parâmetros ajustados para este problema foram:

PARÂMETRO	VALOR
Peso da Função Custo(α1)	1
Peso da Penalidade(α2)	50
Epoch Length	25
Temperatura Inicial	6000
Temperatura Final	20
Constante de Penalidade da Seqüência (α3)	10000
Constante do Makespan(α4)	40
Passo de Movimentação (Δ)	3
Decremento da Temperatura	0,993

Tabela 5.8 : Parâmetros do problema 3

O seguinte procedimento de determinação dos fluxos foi adotado empiricamente:

- f_{ijpq}=1 entre atividades de mesma máquina
- f_{ijpq}=1 entre atividades de mesma ordem
- f_{ijpq}= 0 para outras atividades

Tal escolha de parâmetros de fluxo foi derivada da fase experimental de ajuste, onde notou–se que se os fluxos entre atividades de mesma ordem fossem muito maiores que os fluxos entre atividades de mesma máquina, as atividades de mesma ordem ficavam sempre próximas uma da outra, gerando muita inserção de espera entre atividades de mesma máquina.

O melhor valor encontrado foi o "makespan" 62,5 com um desvio de 13,5% em relação ao ótimo, 55, determinado no trabalho de FISCHER; THOMPSON (1963). A solução, obtida por uma semente inicial aleatória, teve custo total de 5.434,00. O gráfico Gantt resultante desta solução pode ser visualizado na Figura 5.3 e é descrito na Tabela 5.9:

Id	ORDEM (i)	OP (j)	MÁQUINA (k)	NOTAÇÃO	P_{ijk}	S_{ijk}	D_{ijk}
J1	1	1	3	O_{113}	1	18,5	19,5
J2	1	2	1	O_{121}	3	20,5	23,5
J3	1	3	2	O_{132}	6	29,5	35,5
J4	1	4	4	O_{144}	7	39,5	46,5
J5	1	5	6	O_{156}	3	49	52
J6	**1**	**6**	**5**	**O_{165}**	**6**	**56,5**	**62,5**
J7	2	1	2	O_{212}	8	0	8
J8	2	2	3	O_{223}	5	12	17
J9	2	3	5	O_{235}	10	20	30
J10	2	4	6	O_{246}	10	33,5	43,5
J11	2	5	1	O_{251}	10	44	54
J12	2	6	4	O_{264}	4	54,5	58,5
J13	3	1	3	O_{313}	5	1,5	6,5
J14	3	2	4	O_{324}	4	9,5	13,5
J15	3	3	6	O_{336}	8	15	23
J16	3	4	1	O_{341}	9	23,5	32,5
J17	3	5	2	O_{352}	1	39,5	40,5
J18	3	6	5	O_{365}	7	48,5	55,5
J19	4	1	2	O_{412}	5	9	14
J20	4	2	1	O_{421}	5	15	20
J21	4	3	3	O_{433}	5	21	26
J22	4	4	4	O_{444}	3	26,5	29,5
J23	4	5	5	O_{455}	8	30	38
J24	4	6	6	O_{466}	9	53,5	62,5
J25	5	1	3	O_{513}	9	26,5	35,5
J26	5	2	2	O_{522}	3	35,5	38,5
J27	5	3	5	O_{535}	5	39	44
J28	5	4	6	O_{546}	4	44	48
J29	5	5	1	O_{551}	3	55	58
J30	5	6	4	O_{564}	1	60,5	61,5
J31	6	1	2	O_{612}	3	14,5	17,5
J32	6	2	4	O_{624}	3	20,5	23,5
J33	6	3	6	O_{636}	9	23,5	32,5
J34	6	4	1	O_{641}	10	33,5	43,5
J35	6	5	5	O_{655}	4	44	48
J36	6	6	3	O_{663}	1	48,5	49,5

Tabela 5.9 Solução detalhada do problema 3

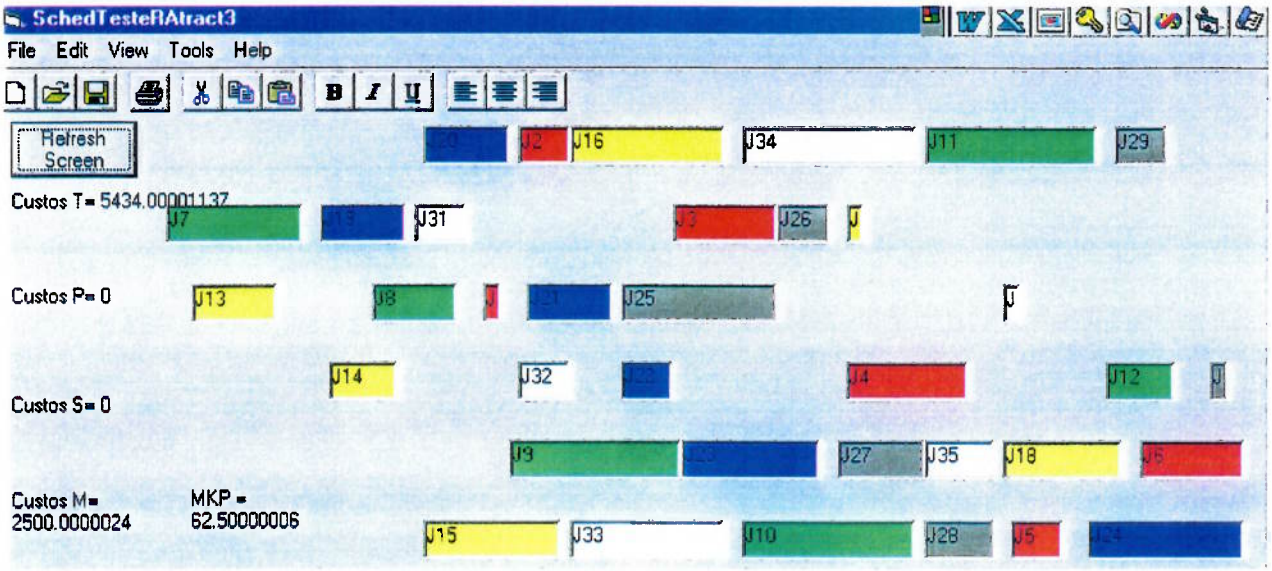

Figura 5.3 Gantt resultante do problema 3

5.2 Problemas de Uma máquina com objetivo de mínimo atraso ponderado (T_{wt})

Para os problemas de atraso ponderado de uma máquina, foram testados 3 problemas, sendo um aleatório e dois extraídos de BAKER (1997). Suas soluções foram comparadas com as regras de despacho EDD, WEDD e SPT. Os resultados destas comparações encontram–se na tabela 5.10.

	Problema 4	Problema 5	Problema 6
Tamanho (n° de ordens)	10	20	20
T_{wt} para EDD	220,00	343,69	345,57
T_{wt} para SWPT	310,00	298,14	346,13
T_{wt} para WEDD	220,00	267,66	293,30
T_{wt} LEO com Solução Inicial Aleatória	210,00	293,37	287,45
T_{wt} LEO com Solução Inicial WEDD	**210,00**	**211,09**	**270,98**
Melhoria sobre EDD	4,55%	38,58%	21,58%
Melhoria sobre SWPT	32,26%	29,20%	21,71%
Melhoria sobre WEDD	4,55%	21,13%	7,61%

Tabela 5.10 Resultados para problemas de uma máquina com objetivo T_{wt}

A. Problema 4

Consiste em um problema de 10 ordens gerado aleatoriamente cujos dados encontram–se na Tabela 5.11.

Id	P_I	d_I	U_i
J1	10	150	1
J2	20	150	1
J3	30	150	1
J4	10	100	1
J5	20	100	1
J6	30	100	1
J7	10	50	1
J8	20	50	1
J9	30	50	1
J10	30	50	1

Tabela 5.11 Dados do problema 4

Os parâmetros utilizados foram:

PARÂMETRO	VALOR
Peso da Penalidade ($\alpha 2$)	85
"Epoch Length"	20
Temperatura Inicial	11000
Temperatura Final	5
Constante do "Tardiness" ($\alpha 4$)	40
Passo de Movimentação (Δ)	3
Decremento da Temperatura	0,99

Tabela 5.12 Parâmetros do problema 4

O gráfico Gantt resultante pode ser visualizado na figura 5.4, cuja seqüência obtida foi (8, 9, 7, 10, 4, 5, 2, 1, 3, 6).

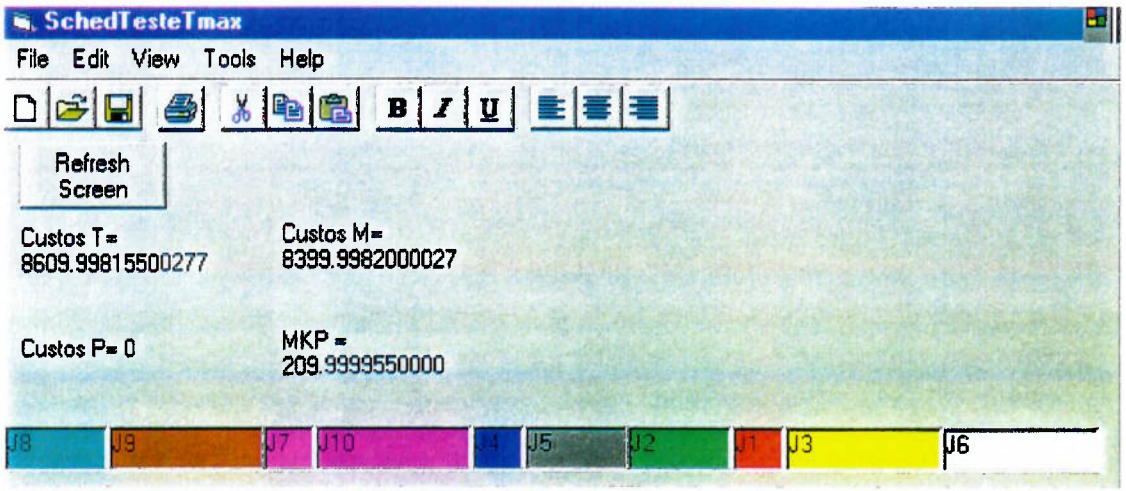

Figura 5.4 Gráfico Gantt para problema 4

B. Problema 5

É um problema de 20 ordens, extraído de BAKER (1997), cada um com seu respectivo peso (U_i), calculado através dos índices de prioridades π_I, onde $U_i=\pi_i/\Sigma\pi_i$

Os dados deste problema podem ser vistos na tabela 5.13.

Id	p_i	D_I	π_i	U_i
1	55	109	10	0,0510
2	68	169	9	0,0459
3	70	1039	7	0,0357
4	73	1158	6	0,0306
5	77	1107	13	0,0663
6	78	0	9	0,0459
7	85	767	12	0,0612
8	86	993	15	0,0765
9	89	643	14	0,0714
10	89	667	10	0,0510
11	92	75	6	0,0306
12	93	612	8	0,0408
13	94	780	12	0,0612
14	94	816	13	0,0663
15	98	721	12	0,0612
16	108	555	5	0,0255
17	126	1166	13	0,0663
18	138	529	8	0,0408
19	143	0	8	0,0408
20	170	1237	6	0,0306

Tabela 5.13 Dados do problema 5

Os parâmetros do método utilizado foram:

PARÂMETRO	VALOR
Peso da Penalidade ($\alpha 2$)	200
"Epoch Length"	25
Temperatura Inicial	2000
Temperatura Final	0,1
Constante do Atraso Ponderado T_{wt} ($\alpha 4$)	40
Passo de Movimentação (Δ)	3
Decremento da Temperatura	0,99

Tabela 5.14 Parâmetros do problema 5

O gráfico Gantt resultante pode ser visualizado na figura 5.5, cuja solução foi a seqüência: (6, 1, 2, 11, 10, 15, 9, 13, 7, 14, 8, 3, 5, 17, 12, 4, 18, 19, 20, 16).

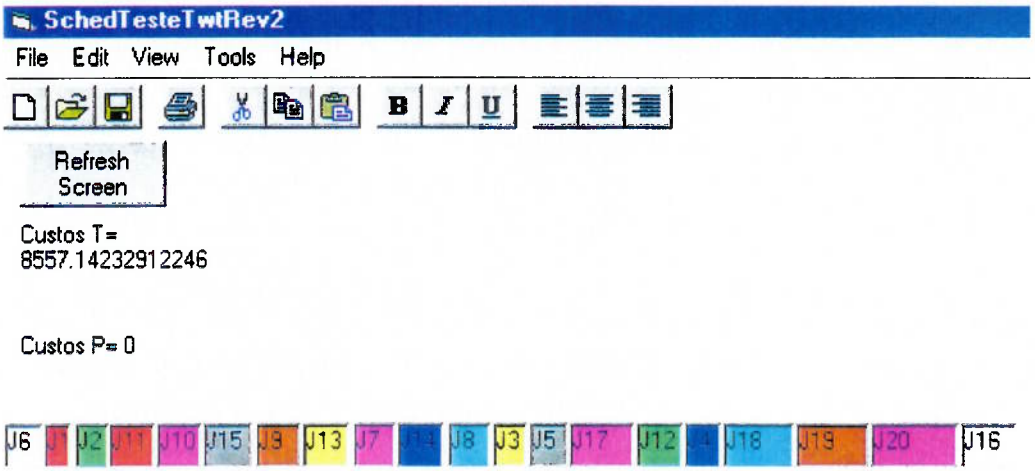

Figura 5.5 Gantt resultante do problema 5

C. Problema 6

Problema análogo ao problema 5. também extraído de Baker (1997) cujos dados se encontram na tabela 5.15

Id	P_i	d_i	π_i	U_I
1	68	437	7	0,03723
2	79	521	5	0,02660
3	80	678	13	0,06915
4	86	841	7	0,03723
5	89	746	6	0,03191
6	94	520	7	0,03723
7	96	610	12	0,06383
8	97	1112	12	0,06383
9	100	772	12	0,06383
10	105	566	8	0,04255
11	106	928	8	0,04255
12	109	472	8	0,04255
13	109	910	12	0,06383
14	112	499	9	0,04787
15	118	498	15	0,07979
16	119	1084	6	0,03191
17	120	617	12	0,06383
18	124	1153	5	0,02660
19	127	1120	14	0,07447
20	135	974	10	0,05319

Tabela 5.15 Dados do problema 6

Os parâmetros do método utilizado foram:

PARÂMETRO	VALOR
Peso da Penalidade(α2)	200
"Epoch Length"	25
Temperatura Inicial	2500
Temperatura Final	1
Constante do Atraso Ponderado T_{wt} (α4)	40
Passo de Movimentação (Δ)	3
Decremento da Temperatura	0,99

Tabela 5.16 Parâmetros do problema 6

O gráfico Gantt resultante pode ser visualizado na figura 5.11, cuja solução foi a seqüência: (12, 15, 17, 1, 14, 7, 3, 9, 4, 13, 8, 11, 19, 10, 6, 20, 5, 16, 2, 18).

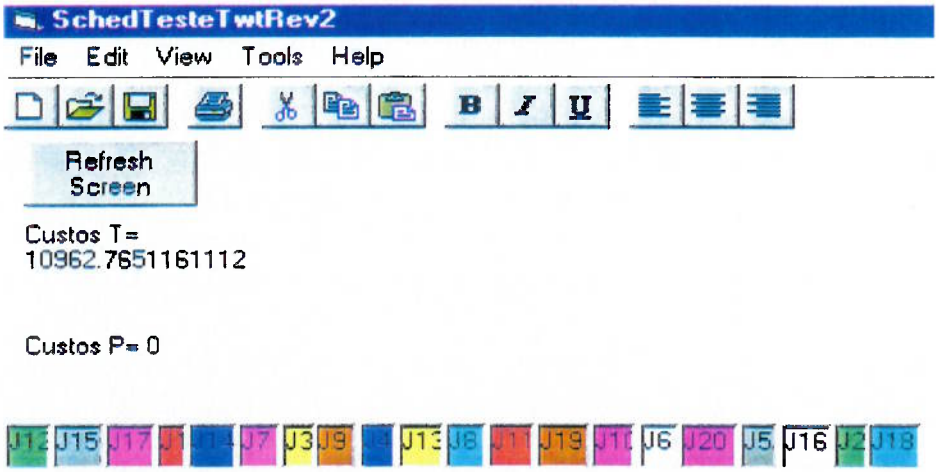

Figura 5.6 Gráfico Gantt resultante do problema 6

5.3 Problemas de Duas máquinas do tipo "Flow Shop" com objetivo de atraso total (T_t)

Para efeitos de teste da performance do algoritmo quanto à produção de soluções exeqüíveis em problemas em que é necessário o respeito à determinada seqüência tecnológica; foram efetuados dois testes adicionais com problemas do tipo "flow-shop" de duas máquinas com complexidade reduzida (respectivamente 4 e 5 ordens). Os resultados resumidos encontram–se na tabela 5.17:

	Problema 7	Problema 8
Tamanho (n° de ordens)	4	5
T_t para EDD	37	30
T_t para SPT	37	33
T_t Ótimo	35	26
T_{wt} LEO com Solução Inicial Aleatória	35	33
T_{wt} LEO com Solução Inicial EDD	**35**	**26**
Observação	Ótimo	Ótimo

Tabela 5.17 Resumo das soluções para problemas "Flow Shop" de duas máquinas

A. Problema 7

Trata–se de problema discutido no artigo de SEN et al.(1989), solucionado então por um método baseado no algoritmo "Branch and Bound". O problema consiste em 4 ordens, processadas em 2 máquinas, gerando um total de 8 atividades. A função objetivo utilizada é "total tardiness" (T_t).Os dados do problema encontram–se discriminados na Tabela 5.18.

Id	ORDEM (i)	OPERAÇÃO (j)	MÁQUINA (k)	NOTAÇÃO	P_{ijk}	d_i
J1	1	1	1	O_{111}	5	10
J2	2	1	1	O_{211}	4	12
J3	3	1	1	O_{311}	10	13
J4	4	1	1	O_{411}	9	15
J5	1	2	2	O_{122}	2	10
J6	2	2	2	O_{222}	5	12
J7	3	2	2	O_{322}	8	13
J8	4	2	2	O_{422}	8	15

Tabela 5.18 Dados do problema 7

Os parâmetros do método utilizado foram:

PARÂMETRO	VALOR
Peso da Função Custo ($\alpha 1$)	1
Peso da Penalidade ($\alpha 2$)	50
Epoch Length	20
Temperatura Inicial	500
Temperatura Final	5
Constante de Penalidade da Seqüência ($\alpha 3$)	10000
Constante da função de Atraso Total T_t ($\alpha 4$)	50
Passo de Movimentação (Δ)	1
Decremento da Temperatura	0,99

Tabela 5.19 Parâmetros do problema 7

O gráfico Gantt resultante pode ser visualizado na Figura 5.7

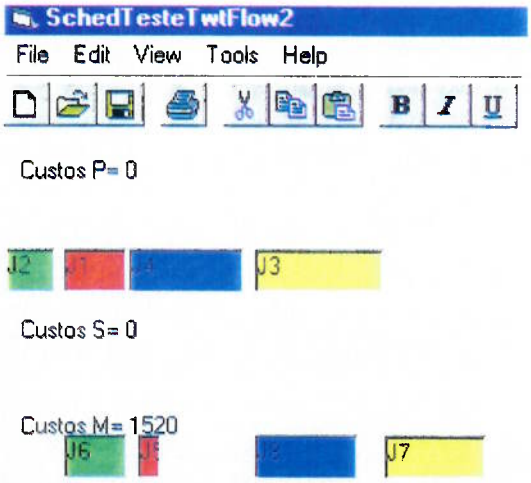

Figura 5.7 Gráfico Gantt resultante do problema 7

A seqüência obtida 2, 1, 4, 3 é a seqüência ótima, segundo trabalho de SEN et al.(1989).

B. Problema 8

Trata–se de problema discutido no artigo de KIM (1993), solucionado então por um método baseado no algoritmo "Branch and Bound". Trata–se de um problema de 5 ordens a serem processadas em 2 máquinas, gerando um total de 10 atividades. A função objetivo utilizada é "total tardiness" (T_t).Os dados do problema encontram–se discriminados na Tabela 5.20.

Id	ORDEM (i)	OPERAÇÃO (j)	MÁQUINA (k)	NOTAÇÃO	P_{ijk}	D_i
J1	1	1	1	O_{111}	6	5
J2	2	1	1	O_{211}	8	7
J3	3	1	1	O_{311}	4	11
J4	4	1	1	O_{411}	5	20
J5	5	1	1	O_{511}	7	36
J6	1	2	2	O_{122}	3	5
J7	2	2	2	O_{222}	4	7
J8	3	2	2	O_{322}	3	11
J9	4	2	2	O_{422}	2	20
J10	5	2	2	O_{522}	6	36

Tabela 5.20 Dados do problema 8

Os parâmetros do método utilizado foram:

PARÂMETRO	VALOR
Peso da Função Custo ($\alpha 1$)	1
Peso da Penalidade ($\alpha 2$)	60
Epoch Length	20
Temperatura Inicial	1500
Temperatura Final	10
Constante de Penalidade da Seqüência ($\alpha 3$)	10000
Constante da função de Atraso Total T_t ($\alpha 4$)	60
Passo de Movimentação (Δ)	1
Decremento da Temperatura	0.99

Tabela 5.21 Parâmetros do problema 8

O gráfico Gantt resultante pode ser visualizado na figura 5.8

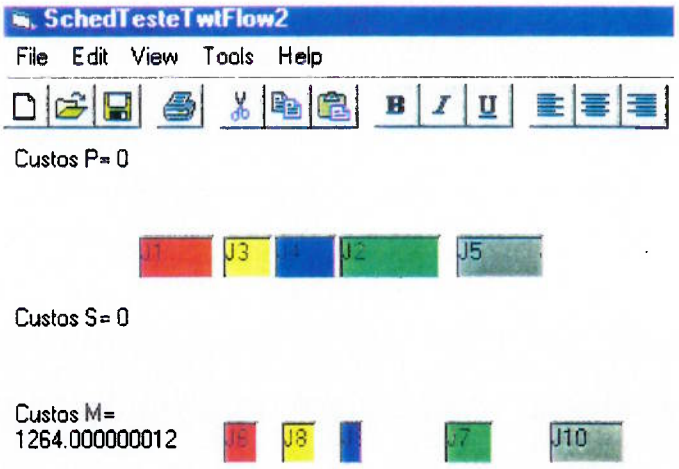

Figura 5.8 Gráfico Gantt resultante do problema 8

A seqüência obtida 1, 3, 4, 2, 5 é ótima, o que pode ser comprovado pelo trabalho de KIM (1993).

5.4 Discussão dos Resultados

5.4.1 Aplicação da metodologia a diferentes configurações de problema

A modelagem foi implementada em três tipos distintos de problemas, diferindo em termos de dimensões (de 6 a 36 operações), na natureza do chão de fábrica ("job-shop", "flow-shop" e única máquina) e nos objetivos ("makespan", atraso ponderado e atraso total).

Os resultados conseguidos atestam a flexibilidade do modelo em conseguir resultados melhores que heurísticas simples para todos os problemas, tendo atingido a seqüência ótima nos quatro casos menos complexos (problemas 1, 2, 7 e 8) e obtidos melhorias significativas que as heurísticas EDD, SWPT e WEDD para problemas de uma máquina com datas (problemas 4, 5 e 6).

5.4.2 Escolha da semente inicial

Os resultados comparativos para problemas de uma máquina entre execuções com solução inicial aleatória e solução inicial por heurística indicam que o algoritmo é sensível à solução inicial. Tal caso torna–se mais evidente quando o algoritmo é parametrizado com um "annealing curto", ou seja, uma relação entre temperatura inicial e final mais baixa (da ordem de 10.000) com um menor coeficiente de resfriamento (menor que 0,8).e "epoch lenght" pequeno (menor que 20), corroborando as observações de POTTS, VAN WASSENHOVE (1991). Contudo, partindo de uma solução inicial às regras SWPT e EDD, o algoritmo convergiu em todos os

experimentos para soluções que obtiveram melhoria em relação à heurística aplicada, indicando robustez.

Para problemas de datas com 2 máquinas (problemas 7 e 8), a convergência para ótimo ocorreu partindo–se de uma semente inicial de regra EDD. Para o problema 7, entretanto, a solução ótima também foi atingida partindo–se de uma solução aleatória.

5.4.3 Convergência do algoritmo e suas soluções

Após a etapa de ajustes de parâmetros, o algoritmo produziu, em cerca de 100% dos experimentos, soluções exeqüíveis ou seja aquelas que respeitam a seqüência tecnológica e cujas atividades não possuem sobreposição. A pior performance neste item foi a do problema 3, de maior complexidade, em que o algoritmo convergiu em 80% dos casos a soluções exeqüíveis. Em todos os outros experimentos, as soluções geradas respeitaram as restrições do problema.

5.4.4 Escolha dos parâmetros

O algoritmo implementado está baseado no "Simulated Annealing" (SA) e portanto seu desempenho está diretamente vinculado à escolha dos parâmetros "temperatura inicial" (T_i), "temperatura final" (T_f), "epoch lenght" (M) e "constante de resfriamento" (r). Além disso, a modelagem possui parâmetros próprios que necessitam ser adequadamente escolhidos paralelamente à escolha dos parâmetros do SA, para que o resultado final convirja para uma solução possível e de desempenho superior às heurísticas mais comuns.

Dentre as observações experimentais, podemos estabelecer algumas diretrizes iniciais para a escolha dos parâmetros:

- Para os problemas T_{wt} e uma máquina, a relação entre o peso de penalidade de sobreposição (α_2) e o peso da função objetivo T_{wt} (α_4) variaram entre 3,25 e 5 para que o algoritmo convergisse para soluções sem sobreposição. Os melhores resultados foram, contudo, atingidos quando a relação estabelecida entre os dois pesos era 5.
- Para problemas de "makespan" para "job-shop" ou "flow-shop", a relação utilizada entre os pesos de penalidade de seqüência (α_3) e o peso da função objetivo de "makespan" (α_4) variou entre 0,8 e 1,25, sendo que os valores mais altos foram utilizados nos problemas de maior complexidade.

O SA convergiu para soluções semelhantes às melhores apresentadas quando a probabilidade de aceitação variou de 60% a 0,01%, probabilidade de aceitação onde, invariavelmente, o SA "congelou".

6 CONCLUSÃO

6.1 Conclusões e contribuições do trabalho

O presente trabalho teve por objetivo desenvolver uma heurística para a resolução do problema de "scheduling" através do uso do "simulated annealing" (SA) aproveitando–se de uma analogia com o problema de arranjo de atividades dentro de um espaço físico (problema de "layout"). Devido à modelagem adotada, o algoritmo produz resultados na forma de gráfico Gantt, ferramenta utilizada para programação de produção nas indústrias.

A abordagem utilizada mostrou–se suficientemente flexível para resolver algumas classes de problemas de "scheduling" como:

- "Job–shop" com o objetivo de minimizar o "makespan" (C_{max})
- "Flow–shop" com o objetivo de minimizar o "makespan" (C_{max})
- Única máquina com o objetivo de minimizar o atraso ponderado (T_{wt})
- "Flow-shop" de duas máquinas com o objetivo de minimizar o atraso total (T_t)

Para os problemas analisados, os resultados alcançados foram superiores a regras freqüentemente utilizadas como SWPT e EDD.

A modelagem é ainda facilmente adaptável para incluir outros objetivos freqüentemente encontrados em literatura como número ponderado de ordens em atraso (N_{wt}), tempo de fluxo ponderado (C_{wt}) e outros, podendo converter–se futuramente em um algoritmo de aplicação geral.

6.2 Sugestões para trabalhos futuros

Tendo como ponto de partida do presente trabalho, podem ser delineadas algumas alternativas de pesquisa, abrangendo duas linhas mestras:

- modelagem mais realística do problema de "scheduling", e
- desenvolvimento da heurística e algoritmos de modo a maximizar a relação tempo de processamento e qualidade da solução final.

Dentro da primeira linha, os trabalhos futuros poderiam seguir os seguintes rumos:

- modelagem de eventos como manutenção preventiva e quebras de máquinas através de inserção de atividades fixas posicionadas dentro do horizonte de tempo da programação;
- possibilidade de preempção;
- adaptação da metodologia para problemas de "scheduling" dinâmicos.

Dentro da segunda linha, podem ser derivados os seguintes trabalhos:

- Estudo da otimização dos parâmetros do SA de modo a obter um melhor desempenho computacional;
- Estudar uma metodologia de arranjo ótimo dos demais parâmetros do modelo de modo a maximizar a convergência e performance do algoritmo;
- Estudo de mecanismos alternativos de geração de vizinhança para o SA.

7 BIBLIOGRAFIA

1. ABRAMSON, D. Constructing school timetables using simulated annealing: sequential and parallel algorithms. **Management Science** 37, n. 1, p. 98–113, 1991
2. ABREU, LUIS F. P. Planejamento e Programação Contínua: Análise da Iteração. **Tese de Mestrado – Escola Politécnica da USP** 1996
3. ADAMS, J.; BALLAS, E.; ZAWACK, D. The shifting bottleneck procedure for job shop scheduling. **Management Science,** v. 34, n.3, março 1988
4. AKPAN, EDEM O. P. Job–shop sequencing problems via network scheduling technique. **International Journal of Operations and Production Management**, v. 16, n.3, p. 77–86, 1995
5. ANTHONY, ROBERT N. Planning and Control Systems: A framework of analysis, Graduate School of Business Administration, Harvard University, Boston, 1965
6. ARMOUR, GORDON C; BUFFA, ELWOOD S. A heuristic algorithm and simulation approach to relative location of facilities. **Management Science,** v.9, n.2 p. 294–300, 1963
7. BAKER, KENNETH R. **Introduction to Sequencing and Scheduling**. John Willey & Sons, New York, 1974
8. BAKER, KENNETH R. **Elements of Sequencing and Scheduling**. John Willey & Sons, New York, 1997
9. BAKER, K. R. Sequence Rules and Due Date Assignments in a Job Shop, **Management Science,** v. 30, p.1093–1104, 1984
10. BARNES, J.; LAGUNA, M. Solving the Multiple Machine weighted flow time using tabu search, **IIE Transactions**, v. 25, n.2, 1993

11. BARRETO, L. W. B. Sequenciamento da Produção através de redes neurais "Self–Organizing". **Dissertação de Mestrado,** Escola Politécnica da Universidade de São Paulo, 2000

12. BARRETTO, M.; CHWIFF, L.; BRITO JR., N.; MOSCATO, L. Scheduling with the LEO Algorithm. **SME Technical Papers**, n. MM98–120, 1998

13. BARRETTO, M.; CHWIFF, L.; BRITO JR., N. LEO Algorithm: a layout based framework for scheduling problems. **Proceedings of ASI 2000**, 2000

14. BOWMAN, E. H. Consistency and Optimality in Managerial Decision Making, **Operations Research**, v.6, n.2, 1963

15. BRANDIMARTE, PAOLO. Neighborhood search based optimization algorithms for production scheduling: a survey. **Computer Integrated Manufacturing Systems** , v.5, n.2, maio 1992

16. BRUSCO, MICHAEL J.; JACOBS, LARRY W. A Simulated Annealing approach to the solution of Flexible Labour Scheduling Problems. **Journal of Operations Research Society** 44, n. 12, p 1191–1200, 1993

17. BUFFA, E.; MILLER. **Production and Inventory Systems: Planning and Control**, Irwin, Homewood, Illinois, 1979

18. BYRNE, M. D., BAKIR, M. A. An application of the multi–stage Monte Carlo optimization algorithm to aggregate production planning. **International Journal of Production economics ,** n.35, p.207–213, 1994

19. CARLIER, J. PINSON, F. E. An Algorithm for Solving Job Shop Problem, **Management Science**, v. 35, p.164–176, 1989

20. CERNY, V. A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. **Journal of Optimization theory applications**. n.45, p.41–51, 1985

21. CHANG, Y. L.; MATSUO, H.; SULLIVAN R. S. A bottleneck based beam search for job scheduling in a flexible manufacturing system. **International Journal of Production Research**, v.27, n.1, p.1949–1961, 1989

22. CHEH, K.; GOLDBERG, J.; ASKIN, R. A note on effect of neighborhood structure in simulated annealing. **Computer Operations Research** v.18 n.6, p.537–547, 1991

23. CHWIF, LEONARDO. Uma metodologia para auxiliar o planejamento do arranjo físico em indústrias de manufatura: base para sistemas computadorizados. **Dissertação de Mestrado,** Escola Politécnica da Universidade de São Paulo 1994

24. CONNOLY, D. T. An improved annealing scheme for QAP. **European Journal of Production Research,** v.57, n.2, p.145–161, 1990

25. CONWAY, R.; MAXWELL, W.; MILLER, L. **Theory of Scheduling.** Addison Wesley Publishing Company, 1967

26. DANNENBRING, DAVID G. An Evaluation of Flow Shop sequencing heuristics, **Management Science** v.23, n.11, p.1174–1182, 1977

27. DELLA CROCE, F.; TADEI R.; VOLTA G. A Genetic Algorithm for the Job Shop Scheduling Problem. **D.A.I**. Politecnico de Torino, Italy, 1992

28. DELL'AMICO, M.; TRUBIAN M. **Applying Tabu Search to the Job Shop Scheduling Problem**, Politecnico de Milano, Italy, 1991

29. DORNDOFF, ULRICH; PESCH, ERWIN. Evolution based learning in job shop scheduling environment. **Computer Operations Research** v.22, n.1, p 25–40, 1995

30. DUDEK, R. A; PANWALKAR, S; SMITH, M. L. The lessons of flowshop scheduling research. **Operations Research** , v. 40, n.1, p.7–13, 1992

31. EGLESE, R. W.; RAND, G. K. Conference Seminar Timetabling. **Journal of Operations Research Society**, v.38, n.7, p.591–598, 1987

32. FISHER, H., THOMPSON, G.L. Probabilistic learning combinations of local job–shop scheduling rules. **Industrial Scheduling,** p.225–251, Prentice Hall, Englewood Cliffs, New Jersey, 1963

33. FOX , M. S. Constraint Directed Search: A case study of Job Shop Scheduling. **Phd Thesis**, Carneggie Mellon University, 1983

34. GILMORE P.C. Optimal and suboptimal algorithms for the quadratic assignment problem. **Journal of the society for Industrial and Applied Mathematics**, v. 10, p.305–313, 1962

35. GLOVER, F. Tabu Search: A Tutorial. **Interfaces**, v.20, n.4, p.74–94, 1990.

36. GOLDBERG. **Genetic Algorithms in Search, Optimization and Machine Learning.** Addison Wesley Publishing Company, 1989

37. GOLDRATT, E. M. **What is This Thing Called Theory of Constraints?,** Mildford, CT: Noth River Press, 1989

38. GREENBERG, H. A branch and bound solution to the General Scheduling Problem. **Operations Research** v.16, p.353–361, 1968

39. HAX, A. C; CANDEA, D. **Production and Inventory Management**. Prentice Hall, Inc, Englewood Cliffs, New Jersey, 1984

40. HILL, CHARLES; JONES, GARRETH. **Strategic Management: an integrated approach**. Houghton Mifflin, 1992

41. HOLT, C. C.; MODIGLIANI, F.; MUTH, J. F.; SIMON, H. A. **Planning Production, Inventories and Work Force**. Prentice–Hall, Englewood Cliffs, New Jersey, 1960

42. IGNALL, E.; SCHRAGE, L. E. Application of the branch and bound techinique to some Flow Shop Scheduling Problems, **Operations Research,** v.13 n.3, 1965

43. JAJODIA, S.; MINIS, I.; HARHALAKIS, G.; PROTH, J. CLASS: Computerized Layout Solutions using Simulated Annealing**. International Journal of Production Research** v.30, n.1, p.95–108, 1991

44. JOHNSON, D.S.; ARAGON, C. S.; MCGEOCH, L. A; SCHEVON, C. Optimization by Simulated Annealing: an experimental evaluation: Part I Graph Partitioning. **Operations Research** v.37, n.6, p.865–992, 1989

45. JOHNSON, D.S.; ARAGON, C. S.; MCGEOCH, L. A; SCHEVON, C. Optimization by Simulated Annealing: an experimental evaluation: Part II Graph Coloring and Number Partitioning. **Operations Research**, v.39, n.3, p.378–406, 1991

46. JOHNSON, S. M. Optimal Two and Three Stage Production Schedules with setup times included. **Naval Research Logistics Quaterly**, v.1, n.1, 1954

47. JONES, C. H. Parametric Production Planning. **Management Science**, v.13, n.11, jul. 1967

48. KARP, R. M. On the Computational Complexity of Combinatorial Problems. **Networks**, 5, p.45–68, 1975

49. KIM, J.; KIM Y. D. Simulated Annealing and genetic algorithms for scheduling purposes with multi level product structure. **Computer Operations Research,** v.23 n.9, p.857–868, 1996

50. KIM, Y. D., Heuristics for flowshop scheduling problems minimizing mean tardiness, **Computer Operations Research,** v. 20, n.4, p.391–401, 1993

51. KIM, Y. D., A new branch and bound algorithm for minimizing mean tardiness in two flow shops, **Journal Of Production Research,** v.44, n.1, p.19–28, 1993

52. KIRKPATRICK, S.; GELATT, C.D.; VECCHI, M. P., Optimization by simulated annealing, **Science** v.220, n.671, 1983

53. KOULAMAS, C; ANTONY, S. R.; JAEN, R. A survey of simulated annealing applications to operations research problems. **Omega, International Journal of Management Science**, v.22, n.1, p.41–56, 1994

54. KOUVELIS, P.; CHIANG, W., – A simulated annealing procedure for single row layout problems in flexible manufacturing systems. **International Journal of Production Research**, n.30, p.7176–7232, 1992

55. KRISHNA, K.; GANESHAN, K.; RAM; JANAKI, D. Distributed Simulated Annealing Algorithms for Job Shop Scheduling. **IEEE Transactions on Systems, Man and Cybernetics,** v.25, n.7, julho 1995

56. LOMINICK Z. A. A branch and bound algorithm for the exact solution of the three machine scheduling problem. **Operations Research Quaterly,** v.1, p.89–100, 1965

57. MACCARTHY, B. L.; LIU, J. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research,** v.31, n.1, p.59–79, 1993

58. MCKAY, K.; SAFAYENI, F.; BUZACOTT, J. Job Shop Theory: What is Relevant? **Interfaces,** , v.18, p.84–90, 1988

59. MAMALIS, A. G.; MALAGARDIS, I. Determination of due dates in job shop scheduling by simulated annealing. **Computer Integrated Manufacturing Systems,** v.9, n.2, p.65–72, 1996

60. MELLICHAMP, J. M.; LOVE R. M. Production Switching Heuristics for the aggregate planning problem. **Management Science**, n.24, p.1242–1251, 1978

61. METROPOLIS, N. Equation of state calculations by fast computing machines, **Journal of Chemical Physics,** n.21, p.1087–1092, 1953

62. MITTHENTHAL, J.; RAGHAVACHARI, M.; RANA, A. I. A hybrid simulated annealing approach for single machine scheduling problems with non regular penalty functions. **Computer Operations Research,** v.20, n.2, p.103–111, 1996

63. MORTON, T.E.; PENTICO. **Heuristic Scheduling Systems.** John Willey & Sons, New York, 1993

64. MUTHER, R. **Planejamento do Layout**: **Sistema SLP.** São Paulo, Edgard Blucher, 1978

65. NAM, S.J.; LOGENDRAN, R. Modified production switching for aggregate production planning. **Computer Operatios Research,** v.22, n.5, p.531–541, 1995

66. OGBU, F.A.; SMITH, D.K. The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem. **Computer Operations Research,** v.17, n.3, p.243–253, 1990

67. OW, P. S.; MORTON, T.E. The single machine early/tardy problem. **Management Science**, v.35, n.2, p.177–191, 1989

68. OW, P. S.; MORTON, T.E. Filtered Beam search in scheduling. **International Journal of Production Research**, v.26, n.1, p.35–62, 1988

69. PACHECO, R., F.; SANTORO, M. C. Proposta de classificação hierarquizada dos modelos de solução para os problemas de job shop scheduling. **Gestão e Produção** v. 6, n.1, p. 1–15, abr. 1999

70. PANWALKER, S.S.; ISLANDER, W. A survey of scheduling rules. **Operations Research** n.25, p.45–61, 1977

71. PARK, M.; KIM Y. A systematic procedure for setting parameters in simulated annealing algorithms. **Computers Operations Research,** v.25, n.3, p.207–217, 1998

72. PINEDO, M. **Scheduling: theory, algorithms and systems**. Prentice Hall, New Jersey, 1995

73. POTTS, C. N.; VAN WASSENHOVE, L.N. Single Machine Tardiness Heuristics. **IIE Transactions,** v.23, n.4, dez 1991

74. PUNNEN, A.P.; ANEJA, Y.P. Categorized Assignment Scheduling: a Tabu Search Approach. **Journal of Operations Research Society**, v.44, n.7, p.673–679, 1993

75. RAGHU, T. S.; RAJENDRAN, C. Due–Date setting methodologies based on simulated annealing: an experimental study in a real–life job shop. **International Journal of Production Research**, v.33, n.9, p.2535–2554, 1995

76. RAMASESH, R. Dynamic Job–Shop Scheduling: A survey of simulation research. **Omega**, v.18, p.43–57, 1990

77. REEVES, C.R. A genetic algorithm for flowshop sequencing. **Computers Operations Research**, v.22, n.1, p.5–13, 1995

78. RONCONI, D. P. Uma contribuição para o estudo do problema de flowshop com buffer ilimitado e zero para minimizar a soma dos atrasos. **Tese de Doutorado**, Faculdade de Engenharia Elétrica e de Computação da Unicamp, 1997

79. SEN, T.; PARTHASARATI, D.; GUPTA, J. N. D. The two flowshop scheduling problem with total tardiness. **Computers Operations Research,** v.16, n.4, p.333–340, 1989

80. SKORIN–KAPOV, J. ; VAKHARIA, A.J. Scheduling a flow–line manufacturing cell: a tabu search approach. **International Journal of Production Research,** v.31, n.7, p.1721–1734, 1993

81. SRINIVISAN, V. A hybrid Algorithm for the One Machine Sequencing Problem to minimize total tardiness. **Naval Research Logistic Quarterly**, v.18, n.3, 1971

82. STOOP, P.; WIERS, V. The complexity of scheduling in practice. **International Journal Of Operations and Production Management,** v.16, n.10, p.37–53, 1995

83. TAM K.Y.; LI S.G. A hierarchical approach to the facility layout problem. **International Journal Of Production Research,** v.29, n.1, p.165–184, 1991

84. TAM K.Y.; LI S.G. A simulated annealing algorithm for allocating space to manufacturing cells. **International Journal Of Production Research,** v.30, n.1, p.63–87, 1992

85. TAUBERT, W.H. A search decision rule for the Aggregate Scheduling Pattern. **Management Science**, v.14, No. 6, p. 343–359, fev 1968

86. TEDESCHI, S. Seleção de Metas–Regras para Alteração Dinâmica do Despacho da Produção. **Tese de Doutorado,** Escola Politécnica da USP, 1996

87. THOMPSON, G. M. A simulated annealing heuristic for shift scheduling using non–continuously available employees. **Computer Operations Research,** v.23, n.3, p.275–288, 1996

88. TSUBONE, H.; ANZAI, M.; SUGAWARA, M.; MATSUURA, H. An interactive production planning and scheduling system for a flow type manufacturing process. **International Journal of Production Economics**, n.22, p.43–51, 1991

89. VAN DONK, D.P., VAN DAM, P. Structuring Complexity in Scheduling: a study in a food processing industry. **International Journal Of Operations and Production Management**, v.16, n.5, p.54–63, 1995

90. VAN LAARHOVEN, P.; AARTS, E.; LENSTRA, J. Job Shop scheduling by simulated annealing. **Operations Research**, n.40, p.113–125, 1992

91. WERNER , F. On the heuristic solution of the permutation flow shop problem by path algorithms. **Computer Operations Research,** v.20, n.7, p.707–722, 1993

92. WIEN, L.M.; CHEVALIER, P.B. A broader view of Job Shop Scheduling problem. **Management Science**, v.38, p.1018–1033, 1992

93. WILHELM, M.R.; WARD, T.L. Solving quadratic assignment problems using Simulated Annealing. **IIE Transactions**, March 1987

94. WRIGHT, M.B. Applying Stochastic Algorithms to a Locomotive Scheduling Problem. **Journal of Operations Research Society**, v.40, n.2, p.187–192, 1989

95. ZEGORDI, S.H., ITOH, K., ENKAWA, T. A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling problem. **International Journal of Production Research,** v. 33, n.5, p.1449–1466, 1995