UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Projeto de Desenvolvimento de um Sistema de Planejamento da Manufatura

Autor : **Carlos Augusto Fernandes Dagnone**
Orientador: **Prof. Dr. Antonio Batocchio**

01/00

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE ENGENHARIA DE FABRICAÇÃO

Projeto de Desenvolvimento de um Sistema de Planejamento da Manufatura

Autor : **Carlos Augusto Fernandes Dagnone**
Orientador: **Prof. Dr. Antonio Batocchio**

Curso: Engenharia Mecânica.
Área de concentração: Materiais e Processos de Fabricação

Dissertação de mestrado apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 2000
S.P. - Brasil

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

DEPARTAMENTO DE ENGENHARIA DE FABRICAÇÃO

DISSERTAÇÃO DE MESTRADO

Projeto de Desenvolvimento de um Sistema de Planejamento da Manufatura

Autor : **Carlos Augusto Fernandes Dagnone**
Orientador: **Prof. Dr. Antonio Batocchio**

_____

Prof. Dr. Prof. Dr. Antonio Batocchio, Presidente
DEF - FEM - UNICAMP

_____

Prof. Dr. Anselmo Eduardo Diniz
DEF - FEM - UNICAMP

_____

Prof. Dr. Arthur José Vieira Porto
USP

Campinas, 21 de fevereiro de 2000

Dedicatória

Nada em minha vida teria sido realizado sem o constante incentivo de minha família. Portanto, a meu pai e à minha mãe dedico mais esta realização.

Agradecimentos

Deixo meus agradecimentos a todos os que me ajudaram ao longo desta caminhada:

Aos doutores do DEF (em especial ao meu orientador Prof. Dr. Antonio Batocchio) e ao doutor da USP (Prof. Arthur José Vieira Porto) que participaram de minha banca;

À Rede de Automação da Manufatura RECOPE / FINEP / BID, pelos recursos computacionais empregados neste trabalho.

Aos amigos de dentro e fora da Unicamp, pelo apoio e pelas horas divertidas;

À CAPES pelo financiamento da bolsa de mestrado;

A todos os funcionários da FEM pela paciência e boa vontade.

“Together we’ll stand; divided we’ll fall”
Pink Floyd – The Wall

Resumo

DAGNONE, Carlos Augusto Fernandes, Projeto de Desenvolvimento de um Sistema de Planejamento da Manufatura, Campinas; Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000. 153 p. Dissertação (Mestrado)

Atualmente, a programação de atividades tem crescido em importância na indústria, a partir do momento em que novos valores foram sendo incorporados aos hábitos de produção. Hoje em dia, um dos objetivos a serem atingidos é o rápido envio de pedidos, ao mesmo tempo em que as datas limites são cumpridas. Sob este ponto de vista, não apenas a programação de atividades é importante para incrementar as capacidades competitivas de uma empresa, mas também introduz uma nova filosofia de produção, baseada na distribuição eficiente de recursos. Este trabalho pretende discutir alguns métodos heurísticos de sequenciamento de atividades por meio de sua implementação computacional via uma linguagem de programação. Tais métodos foram empregados para a criação do aplicativo **SIPMA** (*Sistema Integrado de Planejamento da Manufatura e Automação*), cuja função é fornecer alternativas de soluções para um problema modelado por conceitos de Sistemas Flexíveis de Manufatura (SFM), para o qual um laboratório protótipo localizado na UNICAMP (Faculdade de Engenharia Mecânica – FEM) foi usado. Uma simulação foi feita e seus resultados analisados, o que garante novas direções de pesquisa em termos de plantas mais complexas.

Palavras-chave: Programação de atividades, Métodos heurísticos, Sistemas flexíveis de manufatura, Manufatura integrada por computador.

Abstract

DAGNONE, Carlos Augusto Fernandes, Projeto de Desenvolvimento de um Sistema de Gerenciamento da Manufatura, Campinas; Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000. 153 p. Dissertação (Mestrado)

Recently, scheduling has growing in importance in industry, since new values were added to production habits. Today, a goal for most companies is a high-speed order delivery, with its due dates being fulfilled whenever is possible. Under this point of view, not only scheduling is important to increase enterprise's competitive capacities, but it also introduces a new production philosophy, based on efficient resource distribution for all jobs being processed, minimizing the extras costs derived from its bad utilization. This work is intended to discuss some heuristic scheduling methods through its computational implementation via a computer programming language. They were employed to build **SIPMA** (*Automation and Manufacturing Integrated Planning System*), a software tool whose function is to give solution directions for a problem modeled by FMS concepts, for which a prototype laboratory located at UNICAMP (*Faculty of Mechanical Engineering* – FEM) was used. A simulation was done and its results were analyzed, which will ensure new research directions for more complex plants.

Keywords: Scheduling, Heuristic methods, Flexible Manufacturing Systems, Computer-Integrated Manufacturing

Índice

Lista de figuras

Lista de tabelas

Nomenclatura

Letras latinas

m – número de máquinas

n – número de tarefas

t – tempo corrente

t_i – tempo inicial do horizonte

t_f – tempo de fim de horizonte

p_j – tempo de processamento da atividade j

w_j , w_{Tj}, w_{Ej}, w_{Nj} – peso das atividades

r_j – disponibilidade da atividade j (*ready date*)

d_j – data limite da atividade j

C_j – tempo de completamento da atividade j

C_{max} – função objetivo *makespan*

D – efeito da troca unitária entre duas atividades

F – fator de utilização de um certo recurso

F_j – fluxo da atividade j

F_{max} – função objetivo de fluxo máximo

F_{wt} – função objetivo de fluxo ponderado

L_j – atraso da atividade j

L_{max} – função objetivo de atraso máximo

L_{wt} – função objetivo de atraso ponderado

T_j – adiamento da atividade j

T_{max} – função objetivo de adiamento máximo

T_{wt} – função objetivo de adiamento ponderado

E_j – adiantamento da atividade j

N_{wt} – número de atividade adiadas

ET_{wt} – adiantamento + adiamento ponderados

Z – valor da função objetivo

f – função objetivo

$iR(t)j$ – atividade i tem preferência sobre a atividade j no tempo t

$R(t)$ – custo de uso do recurso no tempo t

I – custo de capital da empresa

$S_j(t)$ – fator de folga da atividade j em relação ao tempo t

$U_j(t)$ – urgência da atividade j no tempo t

Letras gregas

$\pi(i,t)$ – prioridade da atividade i no tempo t

$\pi_{\text{j-ocioso}}$ – prioridade corrigida para o caso de ociosidade inserida

$\delta(x)$ – função discriminante: $\begin{cases} = 1 \text{ para } x > 0 \\ = 0 \text{ para } x < 0 \end{cases}$

Δ - atraso na decisão de programação

Subscritos

i,j – índices das atividades

s – início de período de tempo

f – início de final de tempo

max – máximo ou maximização

wt – ponderado

T_j – relacionado ao adiamento da atividade j

E_j – relacionado ao adiantamento da atividade *j*

N_j – relacionado ao número de atividades adiadas

Siglas

CAF – Custo ao Fim (regra de preemptividade)

CE – Custo por Extensão (regra de preemptividade)

CEM – Custo por Extensão Modificado (regra de preemptividade)

DEF – Departamento de Engenharia de Fabricação

DLMA – Data Limite Mais Adiantada (regra de decisão)

DLMAPD – Data Limite Mais Adiantada Preemptiva Dinâmica (regra de decisão)

EDD – *Earliest Due Date* (regra de decisão)

FEM – Faculdade de Engenharia Mecânica

LMA – Laboratório de Manufatura Assistida

MTPP – Menor Tempo Ponderado de Processamento (regra de decisão)

MTPPPD – Menor Tempo Ponderado de Processamento Preemptivo Dinâmico (regra de decisão)

PEDD – *Preemptive Dynamic Earliest Due Date* (regra de decisão)

SIPMA – Sistema de Planejamento Integrado de Manufatura e Automação

WSPT – *Weighted Shortest Processing Time* (regra de decisão)

Capítulo 1

Introdução

1.1 Contexto da programação de atividades na indústria

A programação de atividades num chão-de-fábrica tem desempenhado ultimamente papel de importância fundamental na indústria, uma vez que novos valores foram sendo incorporados aos hábitos de produção. Na atualidade, uma das metas a serem atingidas é a rapidez e o cumprimento dos prazos de entrega, visando a satisfação do cliente. Sob esta ótica, não apenas a programação é importante para prover vantagem competitiva à empresa, como também introduz uma nova filosofia de produção, baseada na distribuição eficiente de recursos para as diversas atividades a serem processadas, minimizando assim custos extras inerentes ao mau uso daqueles.

A programação de atividades, basicamente, é uma estratégia de planejamento, e, como tal, apresenta níveis que levam em conta o horizonte de tempo de implementação. Em cada nível, as abordagens teóricas diferem, conforme mostrado na Tabela 1, pois a formulação dos problemas é diferente.

Tabela 1: Classificação dos níveis de programação (Morton, 1993)

Nível	Exemplos de problemas	Horizonte
Planejamento a longo prazo	Expansão de planta, *layout* de planta, *design* de planta	2 – 5 anos
Planejamento a médio prazo	Logística	1 – 2 anos
Planejamento a curto prazo	Planejamento de requerimentos, encomenda da fábrica, determinação de prazos de entrega	3 – 6 meses
Seqüenciamento	Roteamento, *job shop*, balanço de linha de montagem, determinação de lote de processos	2 –6 semanas
Seqüenciamento reativo / controle	Entrega de materiais atrasada, quebra de máquinas	1 – 3 dias

Para compreender as diversas técnicas de programação, deve ser notado que o problema geral pode ser tanto formulado de maneira a ser tratável pelos métodos clássicos de otimização (por meio do equacionamento de restrições, introdução das chamadas funções-objetivo, formação de matrizes de representação nó-arco, no caso de problemas de fluxos em redes) como pode ser também estabelecido a fim de que seja possível aplicar a ele regras simples de decisão, métodos iterativos aproximados ou mesmo ambos. Neste caso, a importância reside na experiência que pode ser obtida por meio da observação do mecanismo da planta, uma vez que as regras a serem aplicadas para a resolução do problema devem ser extraídas diretamente dos seus aspectos funcionais (*lay-out* da planta, tipos de máquinas envolvidas no processo de produção e suas características, entre outros). Deste modo, a interação *sistema-homem-planta* é beneficiada pela exatidão de todas as informações que serão tratadas visando melhorar o critério escolhido (que pode variar desde a minimização do tempo total de uso da planta, até a redução de custos com um

determinado recurso). Tal interação é inexistente nos métodos ditos exatos (aplicados aos problemas formulados matematicamente), o que eleva a importância dos métodos de resolução via regras de decisão, pois o que se observa atualmente é uma crescente necessidade de entrosamento entre todos os níveis da produção. Este é um fato importante cuja presença não é possível evidenciar nos casos onde há a formulação matemática exata, na qual se omite, muitas vezes e por questões de simplificação, diversas variáveis fundamentais para a compreensão de tal entrosamento (Morton, 1993).

1.2 Justificativa

Constituindo problema combinatório clássico de otimização, a programação de atividades mostrou-se uma ferramenta de fundamental importância não só na seara industrial, mas em toda situação em que fosse requerida uma alocação racional de *atividades* que demandassem determinados *recursos*, sob certas condições restritivas. Em um hospital, por exemplo, enfermeiras, leitos e equipamentos médicos poderiam ser encarados como recursos, ao passo que pacientes seriam as atividades e as condições restritivas constituiriam tempo de atendimento, disputa por médicos e assim por diante. Em escolas, a atribuição de professores às disciplinas tendo os horários como restrição, a escala de vôos em aeroportos e ônibus e trens em terminais apropriados configuram, de uma maneira ou de outra, problemas de programação para os quais métodos mais ou menos específicos têm sido pesquisados.

É no período inicial da manufatura modernizada na Europa e Estados Unidos (com a introdução, por exemplo, das linhas de montagem), entretanto, que a programação parece possuir raízes mais aprofundadas e na qual uma grande parte da teoria foi desenvolvida, por pioneiros como **Henry Laurence Gantt** (1861 – 1919). A necessidade de se atribuir atividades à máquinas no sentido de minimizar tempos de entrega de encomendas, diminuindo, com isso, estoques e, consequentemente, gastos tornou obrigatória a racionalização de processos, sob pena de perder competitividade num mercado que começava a crescer.

Óbvio está que conforme os problemas se tornavam mais complexos, mais difícil ficava introduzir as restrições adicionais numa estrutura matemática padrão. Desde os esforços iniciais

de Gantt até essa situação, um grande período de tempo passou: apenas nos primeiros anos da década de 50 os trabalhos em programação passaram a ter maior destaque e, a partir daí, outros aspectos começaram a ser estudados, como as abordagens do problema via programação dinâmica e inteira (Hillier, 1967) e considerações sobre complexidade dos algoritmos (Coffman, 1976). De 1980 em diante, maior atenção também foi dada aos problemas de programação estocásticos (Pinedo, 1995).

Num ambiente de manufatura tal como é conhecido atualmente, a programação aparece como parte de uma estrutura maior, fazendo a interface com diversos outros aspectos práticos da indústria (Figura 1):

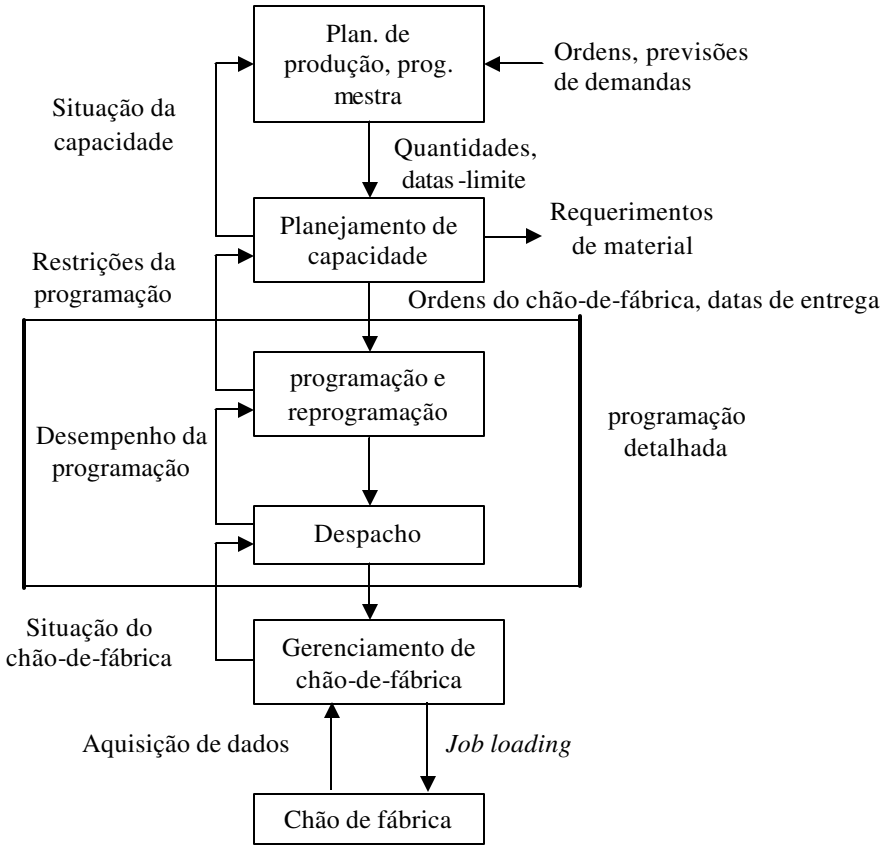


Figura 1 : Diagrama de fluxo de informação num sistema de manufatura (Pinedo, 1995)

Como pode ser visto, a programação obtém informações de outras fontes importantes, principalmente no que diz respeito ao planejamento e controle da produção: leva em consideração

demandas, previsões, níveis de estoque e requerimentos de produtos. Todas as decisões tomadas sobre tais pontos têm impacto sobre o seqüenciamento. Além disso, as próprias decisões tomadas em nível de chão de fábrica podem influenciar o programação e eventos aí ocorridos como falhas e quebras repentinas de máquinas afetam o desempenho do processo.

Judson (1979), por sua vez, propõe a disposição de quatro planos superpostos das principais atividades envolvidas num sistema de manufatura:

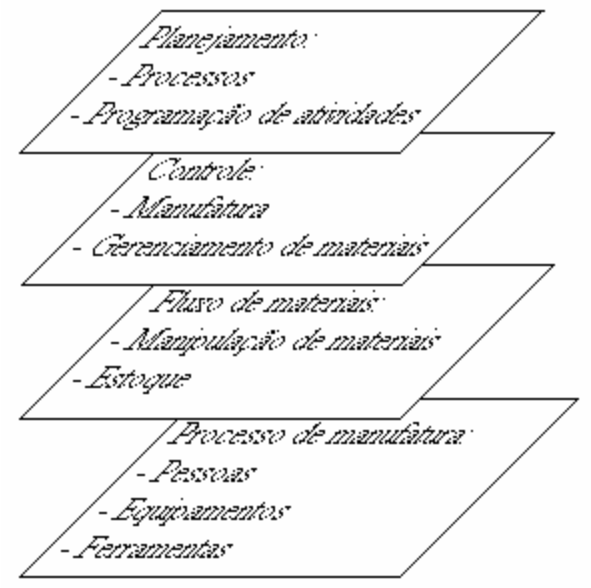


Figura 2 : Conceito de manufatura em quatro planos.

No esquema acima, todas as atividades de planejamento levam à programação de atividades. Isto é feito em diferentes níveis hierárquicos, nos quais a informação processada no nível anterior é refinada até que se transforme em dados úteis, a serem utilizados no nível seguinte.

O nível mais alto é a entrada para o controle: fornece todos os parâmetros para que se iniciem os procedimentos de manufatura propriamente ditos. Aí também se encontram as informações às quais podem ter acesso eventuais rotinas de *feedback*, no caso de problemas (como gargalos) serem encontrados.

Antes do advento da filosofia *just-in-time* (JIT) considerava-se razoável manter estoques relativamente elevados. Entretanto, a manufatura ágil abriu caminho como uma tendência e sua principal característica era a rápida obsolescência dos produtos, aliada à necessidade de uma capacidade de resposta rápida às mudanças que os próprios consumidores demonstravam ao exigirem uma gama de produtos diversificados.

Conclui-se, portanto, ser a programação também uma *estratégia*, que, implementada convenientemente, pode ser o diferencial competitivo para as indústrias dos mais diversos setores.

1.3 Objetivos

Dada a importância da programação de atividades como objeto de incremento da produtividade, diversas soluções computacionais foram propostas. Sistemas como ISIS (Fox e Smith, 1984 *apud* Morton,1993) e CALLISTO (Sathi *et al.*, 1985 *apud* Morton, 1993) foram desenvolvidos e constituem ferramentas de estudos acadêmicos e práticos.

O presente trabalho tem por objetivo os seguintes tópicos:

- Apresentação de uma visão global da programação pela revisão da literatura clássica;
- Desenvolvimento do estudo de uma estratégia de programação, justificando sua escolha em termos de praticidade, facilidade de implementação e eficiência (computacional e prática);
- Criação uma ferramenta computacional, desenvolvida em uma linguagem de programação apropriada, implementando os algoritmos estudados no passo anterior.

Pretende-se contribuir, ao final do trabalho, com uma ferramenta computacional eficiente que possa manipular grandes quantidades de dados e apresentar respostas rápidas para problemas de produção baseados em regras de ordenação de atividades (*scheduling*). Isto será obtido pelo emprego de uma linguagem visual de programação (Borland Delphi 3®), que proverá o usuário de um ambiente gráfico intuitivo, dentro do qual serão passadas as informações principais às heurísticas que serão implementadas, após o que será possível visualizar o resultado em formato

gráfico (cartas de Gantt). Esta conveniência de operação é importante nos sistemas de informação, uma vez que estes devem fornecer informações precisas que reflitam o real nível de desempenho de cada setor da empresa (Certo, 1993). Além disto, a visão objetiva do resultado do problema de programação é uma contribuição importante ao mecanismo de funcionamento do chão-de-fábrica, uma vez que o gerente, analisando as cartas de Gantt, decidirá os rumos de produção ou os adaptará à necessidade presente. Esta interação *sistema-homem-sistema* é de extrema importância e é um campo de estudo ainda incipiente, porém promissor, representado na figura dos *sistemas especialistas*, rapidamente discutidos no próximo capítulo, entre outros aspectos históricos e gerenciais que justificam a classificação do estudo e aplicação dos problemas de programação como **estratégia**, importante metodologia que permeia todos os processos industriais e gerenciais. Também será apresentada no capítulo 2 uma revisão de literatura de programação de atividades, abrangendo tanto métodos exatos quanto os heurísticos (sendo que os últimos terão maior destaque) bem como uma discussão breve sobre células de manufatura e suas aplicações, uma vez que será sobre esta estrutura de manufatura que o sistema de planejamento será concebido.

O capítulo 3 desenvolverá a formalização teórica para a elaboração de heurísticas de programação, baseado na revisão feita no capítulo 2. Serão apresentados os principais resultados (alguns demonstrados) por meio dos conceitos que os envolvem, para o problema de recurso simples / processador simples (RSPS), com disciplinas de chegada estática e dinâmica, cuja aplicabilidade nos Sistemas Flexíveis de Manufatura é direta. No capítulo 4 será considerado um ambiente real de manufatura, a partir do qual serão gerados dados para aplicação direta no SIPMA, um *software* de apoio ao planejamento da produção e que implementa as heurísticas e algoritmos discutidos no capítulo 3. Este *software* fornecerá os resultados da programação de 26 peças hipotéticas idealizadas para a execução de testes, numa célula situada no Laboratório de Usinagem, na FEM – UNICAMP, que é composta por um centro de usinagem, um torno CNC (comando numérico) e uma máquina retificadora.

No capítulo 5 serão apresentados os resultados, seguidos de uma discussão detalhada. Finalmente, será o capítulo 6 o responsável por indicar as direções de trabalhos futuros e as conclusões a respeito dos dados obtidos.

Capítulo 2

Revisão da literatura

2.1 História e revisão da literatura de programação de atividades

Segundo Rembold (1985), foi por volta de 1964 que os computadores começaram a revolucionar a indústria, no setor aeroespacial e automobilístico.

Usados inicialmente apenas para a aquisição de dados, controle de qualidade e monitoramento das funções das máquinas empregadas, foi rapidamente percebido que eles poderiam abranger mais setores da produção, uma vez que as próprias informações básicas adquiridas constituíam os fundamentos da gerência de dados, que poderiam ser transmitidos para os computadores administrativos para posteriores processamentos.

É esta a definição básica de Sistemas de Computação Distribuída para o controle de fábrica.

Concomitantemente, os conceitos de hierarquia em controles computadorizados desenvolveram-se, e um exemplo simples surge diretamente das máquinas CNC: o planejamento e programação são feitos por um computador num nível hierárquico mais alto e a execução das instruções geradas são supervisionadas num nível mais baixo por outro equipamento computacional.

As tecnologias eletrônicas mais recentes permitiram que quase todas as atividades de manufatura fossem controladas, direta ou indiretamente, por máquinas e computadores, o que

gerou uma nova filosofia de produção, denominada Manufatura Integrada por Computador (*Computer-Integrated Manufacturing* – CIM) (Agostinho, 1997), inserida, por sua vez, num contexto mais amplo, o de *Sistemas Flexíveis de Manufatura* (*Flexible Manufacturing Systems* – FMS) que, segundo Ránky (1983), podem ser definidos como "sistemas que trabalham com altos índices de processamento distribuído de dados e fluxo de material automatizado usando máquinas controladas por computadores, células de montagem, robôs industriais, máquinas de inspeção e etc., em conjunto com suporte a material (*material handling*) e armazenamento igualmente computadorizados".

Dentre todos os setores produtivos, um dos mais privilegiados foi o de planejamento, no qual sistemas de programação passaram a ser consideravelmente mais empregados, vistas as facilidades tanto de cálculo quanto de análises rápidas propiciadas por mudanças repentinas no ambiente de fabricação.

Existem diversas aproximações para os problemas de programação, segundo Morton (1993) e Baker (1974). Entretanto, é necessário que seja compreendido, antes, como a programação pode ser empregada eficientemente, de maneira a incrementar o processo produtivo.

2.2 A programação de atividades como ferramenta estratégica

A indústria japonesa deu o exemplo para o resto do mundo em muitas de suas atividades. Principalmente após a II Guerra Mundial (começo da década de 50), houve um crescimento considerável no volume de produção, alicerçado numa pesquisa constante de novos métodos que pudessem auxiliar o país a entrar definitivamente na corrida industrial, após o colapso econômico causado pelo conflito.

A programação de atividades era, até então, um assunto de abordagem puramente acadêmica, pois seu emprego em grande escala era proibitivo devido a fatores simples, como falta de equipamentos computacionais necessários à sua resolução em tempo hábil e de linguagens apropriadas a um tratamento mais simplificado do problema, bem como inexistência, pelos motivos anteriores, de estruturas algorítmicas para o estudo de estratégias de solução.

Buscou-se, portanto, algumas alternativas mais práticas, dentre as quais as filosofias *Just-in-Time* (JIT) e os cartões *Kanban*, que nada mais eram do que soluções intuitivas para o tratamento de problemas de programação simples de fluxo linear. Com isto, pretendia-se minimizar estoques, uma vez que estes representavam custos de manutenção indesejáveis, forçosamente repassados ao preço final do produto. Entretanto, na época posterior a estas aproximações, ainda considerava-se razoável manter tais estoques, tanto de produtos acabados como de peças em processo (*work-in-process* – WIP), para que eventuais erros provocados, principalmente, por demandas mal calculadas impedissem a falta de produtos no mercado, ou causassem problemas de circulação e distribuição de mercadorias.

Ao longo do tempo, com o incremento da competitividade entre indústrias e novos hábitos de consumo, a manutenção de grandes estoques principiou a apresentar problemas mais graves, motivados principalmente por:

- Um significativo aumento de complexidade dos produtos e sua rápida obsolescência;
- Uma necessidade de eliminação de falhas nos processos;
- Uma igual necessidade de se diminuir *lead-times* e tornar possível sua revisão (alteração em alguma ordem de serviço que leva a atrasos ou antecipações programados);
- Necessidades de reações rápidas e flexíveis a mudanças no *mix* de produção e
- Necessidades de medidas de emergência eficientes em caso de problemas no chão-de-fábrica.

Em termos administrativos, a progressiva eliminação de estoques agravava a posição do gerente de produção, que a partir deste momento deveria ter um controle muito maior sobre o mecanismo produtivo, sob pena de perder a idéia global do processo, caso não fosse devidamente entendido.

Felizmente, os computadores acompanharam esta nova "revolução industrial", e a programação de atividades surgiu como opção natural ao dilema da eliminação de estoques, como uma ferramenta auxiliar na organização e manutenção do ciclo produtivo, aumentando assim, o desempenho do sistema como um todo (Figura 3). Equipamentos de custo mais acessível

permitiram que os exercícios acadêmicos de programação saíssem do papel e efetivamente fossem implantados, ainda que de modo incipiente, na indústria.

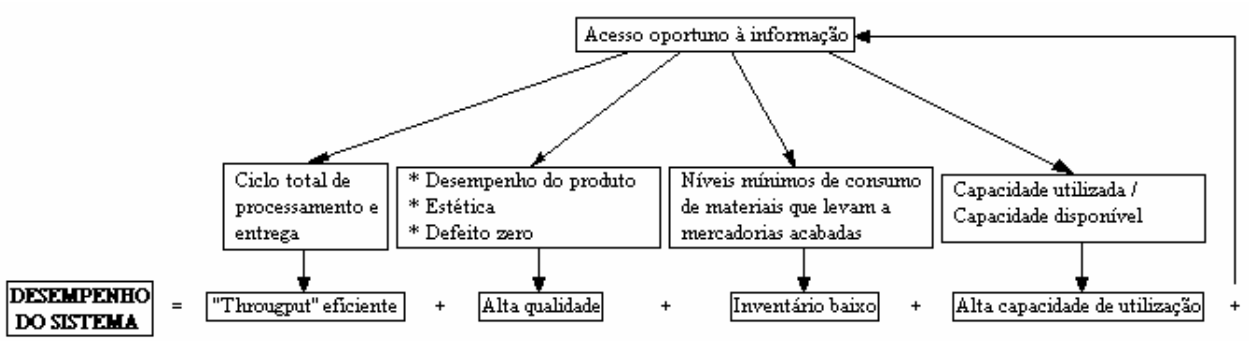


Figura 3: Medidas de desempenho dos sistemas de manufatura (Parsaei, 1990).

A aplicação da programação num ambiente industrial (metodologias, práticas e sistemas de *software*) possui quatro aproximações complementares (Morton, 1993):

- Métodos de treinamento de especialistas humanos;
- *Sistemas especialistas* destinados a imitar as ações humanas;
- Sistemas matemáticos de programação (exatos ou aproximados);
- Sistemas híbridos combinando as vantagens das aproximações anteriores.

A dificuldade principal em se treinar novos gerentes é a complexidade que o chão-de-fábrica pode tomar. Um gerente experiente pode possuir muitos conhecimentos do processo que acompanhou por um longo período de tempo, por exemplo. O que pode ocorrer é a não-difusão deste conhecimento de uma maneira adequada, em pouco tempo, como é o desejável (levando em conta o fator competitividade: substituir o mais rápido possível um funcionário em vias de se aposentar por um novo gerente igualmente qualificado).

A vantagem do ser humano encontra-se em suas capacidades básicas de raciocínio, indução e condensação do conhecimento. Tais habilidades são essenciais para que, em conjunto com o aprendizado aproveitado no treinamento, haja sucesso na formação de profissionais capazes de reagir de modo eficiente no caso de problemas. Neste ponto, muitas empresas optam por recrutar pessoas recém-formadas, para que sejam "adaptadas" aos *seus* métodos de produção.

Quanto aos *sistemas especialistas* (Durkin, 1994), constituem outra forma de conjugar as capacidades humanas e as computacionais. Em sua definição mais simples, um sistema especialista deve ser capaz de "observar" as ações humanas (através da análise de um histórico de atividades, armazenado, por exemplo, em algum banco de dados) e, através de regras, inferências e algoritmos de inteligência artificial (IA), extrair informações que permitam ao próprio sistema computacional "agir", numa espécie de "substituição" ao ser humano (obviamente não uma substituição total, mas apenas um *codificador* de instruções para atividades paralelas e simples), como controlador de um processo produtivo. Está claro que tais sistemas estão muito longe de serem perfeitos nesta atividade, e apenas devem ser considerados como um "auxiliar" ao ser humano, pois não é possível simular atividades mentais humanas imprescindíveis ao controle total: geração de novos ambientes para comparação de situações, percepção de mudanças sutis na organização produtiva e capacidade de reagir a estas mudanças. Em particular, quando tais mudanças acontecem e dispõe-se de um sistema especialista que, de algum modo, participe da estrutura de fabricação, é necessário que seus parâmetros sejam alterados pelo analista responsável para que ele acompanhe a nova situação.

Considerando-se, pois, a aproximação matemática (exata ou aproximada) dos modelos de programação (Baker, 1974, Kondili, 1993 e Rich, 1986), percebem-se uma vantagem e uma dificuldade claras: em primeiro lugar, a linguagem matemática permite que um problema seja estabelecido de uma forma completa. Sendo um problema de otimização, a programação deve apresentar uma *função* que deve ser maximizada ou minimizada, sujeita a diversas condições restritivas. Através do estudo minucioso de um ambiente industrial, é possível estabelecer as restrições e escrevê-las matematicamente, bem como a função (chamada de *função objetivo*), que congregará aspectos econômicos, de penalização, entre outros.

A dificuldade inicial surge quando tenta-se resolver o problema nesta forma "fechada". Questões como implementação computacional e resolução em tempo aceitável aparecem imediatamente como obstáculos, pois mesmo computadores modernos não serão capazes de apresentar respostas rapidamente (o conceito de *tempo aceitável* é relativo e depende muito da situação na qual está inserido. Particularmente, um problema de programação que demande entre 1 ou 2 horas de resolução a cada reconfiguração pode ser inaceitável num ambiente que necessite

de respostas na metade do tempo), dada a natureza essencialmente combinatorial da programação de atividades. Por outro lado, uma simplificação do modelo, apesar de tornar mais fácil sua resolução, pode levar a resultados tecnicamente pouco significativos. Além disso, neste formato, torna-se difícil ao gerente ganhar experiência sobre o esquema de funcionamento do chão-de-fábrica no qual está atuando, pois o processo de resolução é puramente matemático, e não leva em consideração mudanças repentinas que possam ocorrer, em cujo caso será necessário readaptar o modelo matemático convenientemente (conforme comentado no capítulo 1: *Introdução*). A abordagem heurística (através de regras de decisão e algoritmos mais simples) vem ao encontro desta necessidade, porém introduz uma simplificação no modelo, razão pela qual deve-se contar com estimativas precisas sobre *o que é* um bom resultado.

O ideal ainda não alcançado é a cooperação entre os especialistas humanos e os sistemas de programação num único sistema cooperativo multi-experiente:

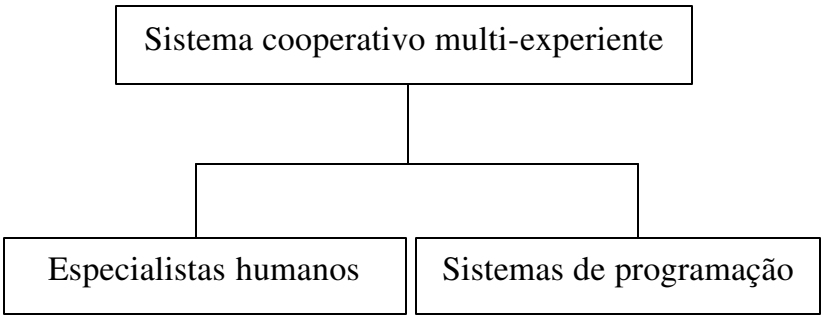


Figura 4: Cooperação entre o ser humano e os sistemas de programação (Morton, 1993).

2.3 A programação de atividades: idéia geral

A programação de atividades é um processo racional de organização, escolha e temporização de uso de diversos recursos para que as atividades desejadas sejam completadas cumprindo certos critérios de custos e, ao mesmo tempo, satisfazendo restrições inerentes ao processo industrial. Assim considerando, a logística, o MRP, *job shops* e linhas de transferência, entre outros podem ser considerados sistemas de programação.

Tais restrições podem ser classificadas, basicamente, em três categorias:

- As que relacionam atividades aos recursos;
- As que relacionam atividades entre si;
- As que relacionam atividades e recursos a eventos externos.

A primeira categoria engloba a designação de atividades aos recursos: qual iniciar primeiro em uma determinada máquina, determinação de tempos de *setup*, entre outras. Já a segunda pode envolver questões como precedência, preemptividade (interrupção) e *delays*. Finalmente, a terceira considera fatores externos como chegada de peças a serem processadas, previsões, resultados gerados nos níveis de produção mais altos (advindos, por exemplo, do MRP), etc.

Como já mencionado, o estabelecimento de um problema de programação em sua forma mais completa pode levar a dificuldades de resolução. Assim, tenta-se resolver primeiro versões mais simples do problema, para que certas sutilezas sejam compreendidas, e praticar uma *análise de sensiblidade*: quão sensível é uma resposta em termos de inclusão de novas restrições ? Deste modo, gradualmente pode-se anotar as diferenças entre problemas mais complexos e deduzir os resultados destes pelos que foram obtidos dos problemas mais simples (Bazaraa, 1990).

As soluções são obtidas pela maximização ou minimização de uma função convenientemente obtida pelo levantamento de informações a respeito do que se quer atingir. Geralmente, deseja-se diminuir os custos de uso de um determinado recurso, diminuindo, por exemplo, seu emprego no processo, através de processos alternativos. Ou pode-se desejar imputar uma penalização por atraso em entregas para um deteminado cliente, que pode figurar na função com um peso negativo que cresce à medida que o prazo não é respeitado. As possibilidades constituem um leque extenso, e isto torna difícil a elaboração de uma função objetivo que atenda necessidades particulares. De um modo mais geral, deseja-se uma função que:

- Maximize a produção em algum dado período de tempo;
- Satisfaça o consumidor quanto à qualidade e à prontidão com a qual o produto é oferecido e

- Minimize os custos de produção.

Algumas idéias adicionais que podem estender o modelo de programação padrão (que foi o descrito até agora) estão presentes na *reconfiguração* e tópicos em *sistemas de informação*. O conceito básico de reconfiguração trata de adaptar o modelo corrente em termos de quantidades produzidas e disposições internas de recursos, para compensar sobrecargas inesperadas provenientes de uma demanda não prevista por determinado produto. Deste modo, a reconfiguração cuidará de ativar e alocar convenientemente, por exemplo, linhas de produção desativadas, máquinas para trabalho paralelo com outras já em funcionamento, ou mesmo contratação temporária de maior número de trabalhadores. Para que isto seja conseguido, os *sistemas de informação* devem prover dados que suportarão este esquema extra de configuração, por meio de previsões internas e externas (baseadas em históricos anteriores, ou em novas demandas atualizadas), agrupamento de atividades / recursos que levarão ao agregamento de tarefas, entre outros.

Neste ponto, torna-se necessário diferenciar os termos *atividade* e *tarefa*, pois certas confusões podem ocorrer. A primeira designará uma operação a ser desempenhada, dentro de um conjunto maior, visando a produção de um determinado bem. Assim, uma *tarefa* (ou um *job*) possuirá um certo número de *atividades* (ou operações) de cuja conclusão depende a apresentação de um produto acabado. Por ocasião da apresentação das abordagens matemáticas no momento oportuno, será empregado, preferencialmente, o par *operações / tarefas* para designar estas funcionalidades.

Outra observação a ser feita é a respeito de agrupamento de atividades e recursos. O primeiro é relativamente simples, e já foi mencionado: todo grupo de atividades que permitam completar uma ordem e gerar um produto é uma tarefa. Esta distinção é necessária porque um grupo de atividades pode representar uma tarefa ou várias. Por outro lado, a distinção entre recurso único e um grupo de recursos é mais complicada. Para esta distinção, levando-se em conta tanto uma máquina como um grupo delas, será considerado *um único* recurso se as únicas decisões a serem tomadas a respeito desta máquina ou deste grupo forem:

- A próxima atividade nele(a) a ser iniciada;
- O tempo em que a atividade deve começar e
- O tempo em que a atividade deve deixar a máquina/grupo (*completion time*).

Para ilustrar o conceito, sejam duas máquinas idênticas. Se houver apenas uma fila para as duas, e a que for liberada primeiro é a que receberá a próxima atividade, então tem-se um caso de recurso único. Se, de modo diferente, cada máquina tem uma fila própria, então tratam-se de recursos distintos.

A seguir serão apresentadas definições breves de alguns ambientes no qual os sistemas de programação podem ser empregados. Estão contidos no quarto nível de programação de atividades (ver Tabela 1, capítulo 1) e foram escolhidos por constituírem as funções mais básicas de um chão-de-fábrica atual. Preferiu-se manter os nomes técnicos em inglês.

FLOW SHOP: regime de produção no qual há um número de tarefas cujas atividades são desempenhadas na mesma ordem, em todas as máquinas (o caso mais simples). O fluxo de atividades pode ser discreto, contínuo ou semicontínuo. Um *flow shop* "puro" é raro; *flow shops* compostos são mais comuns. A composição é representada pela substituição de uma máquina simples por máquinas compostas, em paralelo. Outro fato comum é o *fluxo reentrante*, no qual uma máquina pode ser usada mais de uma vez para a mesma atividade;

JOB SHOP: regimes *job shop* são mais comuns e mais complexos de serem analisados. Diferentemente dos anteriores, o fluxo de atividades pelas máquinas pode ser diferente para cada tarefa. É também chamado de *closed shop* porque geralmente as partes em processo (WIP) de uma tarefa não são aproveitadas em outra e todas as ordens são distintas;

OPEN JOB SHOP: semelhante ao *job shop* um *open job shop* produz para inventários finais, ao invés de produzir para satisfazer ordens diretas;

BATCH SHOP: é um *open shop* adaptado a WIP´s de grandes dimensões, justificando assim a produção em lotes, fazendo uso da economia de escala. É um tipo de regime muito

empregado na indústria química e um exemplo de formulação neste sentido pode ser encontrado em Kondili (1993);

BATCH/FLOW SHOP: é uma composição entre os dois regimes, muito comum também na indústria, principalmente alimentícia, na qual há um processo inicial de lote (elaboração, preparação e cozimento, por exemplo, de um determinado alimento), seguido de um fluxo linear (seu envasamento, que pode passar por diversos processos).

CÉLULAS DE MANUFATURA: as células de manufatura (discutas em detalhe na seção 2.6 deste capítulo) são uma tentativa de conciliar a flexibilidade dos regimes *job shop* com a simplicidade dos regimes *flow shop*. Basicamente, os itens a serem processados no *job shop* são agrupados em famílias servida por uma célula que, em sua composição típica é um condutor (*handler*) automatizado, cercado por máquinas, eletronicamente sincronizadas e programadas via comando numérico (controles CNC). Como as tarefas são semelhantes, é possível agrupar todas as máquinas necessárias para completá-las. Se as máquinas puderem ser acessadas em qualquer ordem com rapidez, a eficiência do *flow shop* se revela e justifica sua aproximação em conjunto com o conceito de *job shop*.

ASSEMBLY SHOP: pode ser um *open shop* ou um *batch shop* que congrega atividades de sub-montagem em diversos níveis para, no último nível, seja possível completar a montagem total de um determinado produto.

LINHAS DE MONTAGEM/LINHAS (FLEXÍVEIS) DE TRANSFERÊNCIA: são versões do *assembly shop* para diversas capacidades. A linha de montagem serve capacidades produtivas médias para altas, com pouca variação nos produtos. Um linha de transferência, por sua vez, serve altas capacidades produtivas, com variação praticamente nula. A linha flexível de transferência também serve altas capacidades, mas pelo fato de empregar toda uma estrutura eletrônica de transporte (o que caracteriza a manufatura flexível) torna-se muito mais fácil obter variação maior no que é produzido. Sob outro ponto de vista, linhas de montagem podem ser vistas como uma *fila seqüencial*, e, portanto, modelada e simulada segundo critérios mais específicos (Papadopoulos, 1993).

Vistos tais conceitos, torna-se claro que cada regime de produção demanda abordagens diferentes se é pretendido que um sistema de programação otimize suas operações. Na realidade, existem algumas poucas aproximações que, modificadas corretamente, podem cobrir a grande maioria dos casos. Podem ser citados:

Programação por intervalo (PI): uma programação formal é dada antecipadamente e espera-se que o processamento atual ajuste-se a esta programação independente de ocorrerem falhas repentinas no sistema. A PI é útil quando se deve gerenciar diversos recursos críticos, mas tem como desvantagem a introdução de *gaps* e outras ineficiências que tendem a tornar a produção instável;

Programação por despacho (PD): neste caso uma programação prévia pode ou não ser dada, e é possível de tornar-se sensível a problemas através de mudanças apropriadas. A estratégia é programar recurso por recurso mantendo-os ocupados com a operação de mais alta prioridade; liberada uma máquina, a próxima operação mais prioritária é programada nela e assim sucessivamente. Uma vantagem principal desta aproximação é a produção de um *seqüenciamento* compacto, logicamente definido em termos de prioridades conhecidas. Por outro lado, uma desvantagem evidente é a incapacidade de se lidar (pelo menos de uma maneira fácil) com qualquer comprometimento de recursos mais elaborado (como, por exemplo, operações preemptivas);

Programação por despacho simples (PDS): é um caso mais geral no qual o anterior está inserido; os recursos nunca são deixados ociosos por antecipação da chegada de tarefas prioritárias. Costuma-se empregar o termo "despacho" como sinônimo para a PDS e os outros casos restantes são ditos de "ociosidade inserida";

Programação de tarefa crítica: todas as operações da tarefa mais crítica são programadas através da planta em primeiro lugar; após, vêm as operações da segunda tarefa mais crítica, até o final. Embora a idéia seja justamente cuidar das tarefas de maior prioridade, dedicando-lhes todos os recursos, as demais, obviamente, poderão não tê-los disponíveis no momento em que forem processadas;

Programação de recurso crítico: o recurso "gargalo" (ou sobrecarregado) é programado primeiro, após o que os demais são programados ao redor deste. A idéia desta aproximação é que muitas tarefas críticas serão beneficiadas pelo uso eficiente do recurso gargalo.

Programação de operação crítica: analisando as tarefas, o par operação/recurso com maior prioridade é programado em primeiro lugar. Este método tenta combinar as vantagens dos dois métodos anteriores.

Programação para frente: podem produzir resultados factíveis e compactos, embora nada garanta que as datas limite (*due dates*) sejam cumpridas. Programações desta natureza costumam ser do tipo despacho, mas também podem ser de intervalo;

Programação para trás: é o programa feito anteriormente às datas-limite. Podem, por isto, satisfazê-las, pagando a pena, entretanto, poderem ser infactíveis. Costumam ser do tipo intervalo, mas também podem ser de natureza de despacho;

Programação heurística de despacho (PHD): um método de programação à frente nos quais em cada ponto de escolha (temporização, roteamento, por exemplo), são calculados índices de prioridade por alguma regra, sendo escolhida aquela que tiver o índice mais relevante, segundo algum critério;

Programação avançado de despacho: é uma PHD que prevê datas-limite e recursos críticos dinamicamente. É o objetivo de estudo deste trabalho e é a aproximação sobre a qual está baseado o *software* de programação proposto;

Programação combinatorial: são métodos que tomam um subconjunto de soluções e o analisa através de uma árvore de busca. Embora busquem por soluções ótimas (e não boas aproximações apenas), torna-se proibitiva sua implementação computacional devido aos altos tempos de processamento que tais rotinas requerem. Podem ser citadas a programação inteira, a programação inteira aproximada e a programação dinâmica como alguns exemplos deste segmento.

Programação exaustiva: método também proibitivo por seu alto custo em termos de tempo. Também busca por soluções ao longo de uma árvore, mas esgota todas as possibilidades de solução em um determinado ramo desta antes de passar a outro.

Segundo Morton (1993), as programações acima podem ser enquadradas nas seguintes categorias:

- Aproximações do tipo manual-intervalo / manual-despacho;
- Aproximações do tipo simulação-despacho;
- Aproximações do tipo matemático-exato / matemático-heurístico;
- Sistemas especialistas;
- Sistemas mistos de Inteligência Artificial / Pesquisa Operacional / Suporte à Decisão.

Em se tratando das abordagens do tipo intervalo, o que se deseja é uma precisa adequação entre recursos e atividades; as do tipo despacho, por sua vez, requerem temporizações exatas, ao mesmo tempo em que se mantém coerente a programação, situação que pode ser tornada computacional através da abordagem simulação-despacho, em suas situações mais simples. Os sistemas matemático-heurísticos (que serão discutidos na seção 2.5: *Aproximações matemático-heurísticas*) têm por objetivo resolver aproximadamente os problemas formulados formalmente, como será visto na seção seguinte. Por fim, os sistemas especialistas (já brevemente mencionados na seção 2.2: *A programação de atividades como ferramenta estratégica*) são a implementação computacional de situações mais complexas de programação tipo manual-despacho e podem ter suas vantagens aproveitadas nos sistemas mistos de IA/PO/SD, que também tentam concatenar as vantagens dos sistemas matemáticos e de suporte à decisão.

2.4 Metodologias de modelagem de problemas de programação

O tratamento de problemas de programação via métodos exatos (principalmente aqueles que se utilizam de programação matemática) requer que sua estrutura seja convenientemente representada em termos de suas principais restrições:

- Como os processadores estão sendo ocupados;
- A ordem que as operações tomarão nesta ocupação e
- A descrição das restrições inerentes ao processo e aos recursos compartilhados.

Durante a modelagem do problema, costuma-se deixar implícita a representação das rotas de processamento nas relações de recorrência que estabelecem os instantes de ocorrência dos eventos. Desta maneira, torna-se claro que quanto mais simples for tal representação, menor é a flexibilidade da estrutura de processamento.

Não se pretende introduzir neste texto formulações completas de problemas de programação, mesmo porque estas variam muito de acordo com a abordagem da situação. Entretanto, é interessante notar certos artifícios que se empregam para tornar as formulações mais fáceis em termos de implementação computacional.

Um tópico de extrema importância na formulação de problemas de programação de atividades é a representação da ocupação dos processadores pelas operações. Os métodos exatos buscam uma solução ótima através da enumeração racional de combinações de alocações; portanto, é necessário que esta ocupação seja devidamente representada no modelo. Um dos métodos desenvolvidos para formulação é o uso de *variáveis binárias*, que impõe uma condição do tipo SIM ou NÃO como resposta, por exemplo, a uma solicitação de alocação de atividade numa determinda máquina, avaliando os valores da função objetivo para as opções disponíveis.

De modo geral, o uso de variáveis binárias para o modelamento de problemas de programação é muito comum. A literatura trata de dois casos principais: aquelas variáveis que indicam a ocupação do processador por uma operação durante um determinado instante de tempo, e as variáveis que indicam o início do processamento de determinada operação num dado processador. Estas, por sua vez, estão englobadas numa categoria maior: a de variáveis binárias de *alocação*. Rich (1986) também considera o uso de variáveis binárias de *ordenamento*, que impõe ordens de precedência entre operações.

Em se tratando de variáveis binárias de alocação, o tempo pode ser tratado de maneira pré-definida na formulação (Kondili, 1993 e Shah, 1993, Grossman, 1992 *apud* Kondili, 1993) ou encarado implicitamente, através do conceito de *slots* (Sahinidis, 1989, Pinto, 1993 *apud* Kondili, 1993).

Um *slot* é um intervalo de tempo dentro do qual as operações devem ser desempenhadas. Sem um tamanho fixo, em princípio, um *slot* terá sua duração definida pelas operações que forem alocadas para o processador em questão. Assim sendo, o número de intervalos de tempo contidos num *slot* é igual ao número de operações que se servem do processador (se não houver máquinas em paralelo).

Kondili, Pantelides e Sargent (1993) propõem uma formulação baseada em um recurso denominado *state-task-netowork*. Nesta aproximação, designada para tratar de problemas de programação de ativiades em lotes (*batch scheduling*), é empregada uma estrutura de representação de fluxo de matérias primas através de diversos estágios. Todo o material pode ser bruto (recém-recebido para processamento), ou já pré-processado num estágio anterior, podendo servir de entrada para outro posterior. Forma-se, assim, uma *receita* que indicará todo o roteiro de processamento do material, até o produto acabado.

A estratégia de uso de um *state-task network* consiste na representação da receita pelo uso de dois ícones: um círculo, pra indicar os estados transientes e retângulos, para indicar as fases de processamento propriamente dito. Um exemplo simples pode ser visto Figura 5.

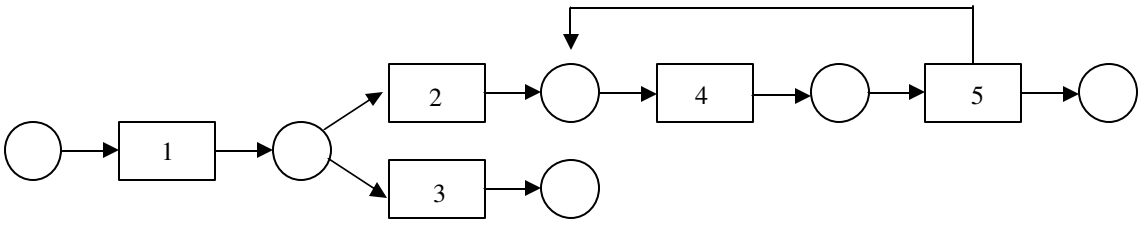


Figura 5: Exemplo simples de uma receita no formato *state-task network* (Kondili, 1993).

Esta é uma representação conveniente, que elimina dúvidas a respeito da ordem de processamento e do fluxo de entrada entrada de materiais.

Com a crescente popularização dos computadores a partir de 1950, nos Estados Unidos, foi dada maior importância à tentativa de desenvolvimento de métodos que pudessem tratar com o problema de programação formulado teoricamente. Em outras palavras, esforços foram realizados no sentido de se programar, efetivamente, os algoritmos empregados na resolução manual de tais problemas (o que, obviamente, dependendo das dimensões do problema, tornava-se uma tarefa impossível). Um dos métodos mais conhecidos é o de *Branch-and-Bound* (B&B) que, conceitualmente, é de fácil compreensão e será rapidamente introduzido.

A técnica B&B consiste numa enumeração controlada de soluções possíveis para o problema de programação. Sejam consideradas 15 tarefas a serem seqüenciadas em uma máquina. Em princípio, há 15 escolhas para a primeira tarefa a ser desempenhada. Após esta escolha, há 14 opções restantes. Logo depois que se escolhe a segunda tarefa, existem ainda 13 opções, até que todas sejam seqüenciadas. Nota-se, portanto, que o número de soluções possíveis é muito grande (15!), mesmo para a pequena quantidade de tarefas. Obviamente, quando se trata de soluções, está-se falando de uma seqüência de tarefas que otimize uma função objetivo dada, e é em termos do valor desta função que é escolhida a seqüência ótima. Entretanto, é possível mostrar que certos ramos desta árvore de soluções podem ser eliminados, por só apresentarem soluções não-ótimas. Aplicando-se estas "eliminações" sucessivamente, chega-se à solução desejada sem a necessidade de se enumerar todas as soluções possíveis.

Apesar de ser um método exato, e, portanto, fornecer uma solução ótima que atenda todas as restrições do problema, o B&B e outros métodos exatos são do tipo *np-completo*, o que significa, grosso modo, que não existe maneira de se resolver tais problemas sem que a dificuldade no emprego de um determinado método cresça exponencialmente com a dimensão deste. Por esta razão, os métodos matemático-heurísticos têm crescido em importância, principalmente porque aliam os rigores matemáticos dos métodos exatos ao bom senso e ao *insight* conseguido através das análises aproximativas que desempenham. Entre tais métodos,

figuram a *busca tabu*, o ***simulated annealing***, os *algoritmos genéticos*, o ***beam search*** e a *dinâmica de gargalos.*

2.5 Aproximações matemático-heurísticas

A grande maioria dos métodos matemático-heurísticos empregados na atualidade fazem uso de uma técnica mais geral denominada *busca em vizinhança*: dada uma solução inicial para um problema de programação de atividades (calculada por qualquer outro método), a idéia é tentar, por todos os meios possíveis, efetuar pequenas mudanças nesta solução (portanto, mantendo-se na *vizinhança* dela) e avaliar sua função objetivo. Se não é possível melhorar o valor da função, então o método está terminado. Se isto é possível, então o método move-se até aquela solução e passa a buscar por melhorias a partir daí, até que um critério seja cumprido. Dois métodos fazem uso desta técnica:

- A *busca tabu* (Glover e Laguna, 1993) é uma variação da busca em vizinhança que mantém uma lista de direções de busca. Após avaliada qual a melhor direção (a que melhora mais significativamente a função objetivo), as demais são bloqueadas, impedindo-se, assim, que o método regrida para soluções piores.

- O ***simulated annealing*** (van Laarhoven e Aarts, 1997), por sua vez, adiciona uma quantidade aleatória a cada provável direção de busca, o que vai gerar uma diferença quando se calcula a função objetivo, em comparação com seu valor atual. Esta diferença entre o valor atual e os valores obtidos com esta adição aleatória é denominada *diferença ajustada*. Será escolhida, então, a direção que tenha produzido a diferença ajustada mais positiva.

Outra técnica heurística de relevo são os *algoritmos genéticos* (Dasgupta, 1992), baseados no processo de evolução natural: dada uma solução e possíveis direções de busca, as que melhor incrementarem a função objetivo têm a permissão de gerar novas direções, através da incorporação das qualidades dos "pais", enquanto as demais são eliminadas, para manter a "população" estável. Estes algoritmos genéticos podem ser considerados casos abrangentes dos dois métodos anteriores.

Por fim, o ***beam search*** (Zhang, 1998) é um método desenvolvido por especialistas em IA e se destina, de modo semelhante ao B&B, à busca parcial em árvores. A diferença é que, ao invés de descartar *todos* os ramos sem uso, pode desejar eliminar os que são *provavelmente* sem uso. Claro que é necessário estabelecer o que significa *provavelmente inútil*: é um dos pontos de estudo do método e envolve, inclusive, o abandono de ramos que possam levar a caminhos pouco promissores.

2.5.1 A dinâmica de gargalos

O funcionamento da dinâmica de gargalos se dá pela *estimativa de preços* de certas variáveis de interesse no processo produtivo. Duas delas são a base para todas as decisões que serão tomadas depois: o *custo por atraso em execução de tarefas* e o *custo por atraso em uso de recursos*. A análise conjunta da influência de tais custos por atraso permite o cálculo de qualquer índice de decisão para qualquer outro evento que tomará lugar na planta (temporização, roteamento, escolha de seqüência, entre outros). Desta maneira, os preços calculados permitem tomar decisões locais, com cálculos fáceis, e permitem a solução do problema via seqüenciamento avançado de despacho. Esta aproximação que se utiliza de custos por atraso não é a ótima, uma vez que não se pode determinar, com exatidão, qual conjunto de valores produzirá a melhor solução, embora o bom senso e o conhecimento da planta seja essencial para que os resultados obtidos possam ser confiáveis.

Para a estimativa dos custos por atraso em execução de tarefas, um dos fatores que pode ser levado em consideração é a insatisfação do cliente pela demora, que cresce proporcionalmente à importância da tarefa a ser desempenhada e, obviamente, à quantidade de dias de atraso. Cada atividade possui uma data limite e uma tolerância. Se a função objetivo é minimizar a insatisfação do cliente pela demora, então esforços no sentido de se manter a produção dentro da tolerância, ou, pelo menos, minimizar os atrasos, podem ser úteis na estimativa de insatisfação e, desse modo, possibilitam a obtenção dos custos por atraso. Note-se a dificuldade inerente a este processo, uma vez que as datas limite das tarefas ainda não podem ser determinados, uma vez que o seqüenciamento ainda não foi executado. Porém, uma simplificação importante que costuma ser introduzida neste caso é o de *custo constante por atraso*: assim, as datas limite

passam a não mais ter relevância na medida de insatisfação, e podem ser estimadas por diversos métodos (Morton, 1993):

- Múltiplo de um histórico fixo do tempo de processamento restante;
- Iteração por data limite (através de rodadas sucessivas da simulação para refinar este valor);
- Ajuste histórico das datas limite *x* medidas de desempenho;
- Julgamento humano baseado em experiências anteriores.

Como pode ser notado, a dinâmica de gargalos repousa fortemente na observação e anotação dos eventos que ocorrem no chão-de-fábrica, bem como experiências anteriores. Isto também pode ser observado a seguir, quando será tratado da estimativa de custos por atraso em recursos.

Para que sejam estimados os preços por atraso no emprego de máquinas, costuma-se empregar a análise do *período de inatividade* destas. Em outras palavras, seja considerado um processo de programação e uma determinada máquina, que possui uma fila de operações a serem processadas e outras que são sabidas chegarem àquela máquina em certos períodos de tempo. Se este recurso for desativado por um dia, o custo, portanto, será a soma do custo todas estas operações que estão em fila vezes este período de desativação. Não apenas estas, mas também todas as atividades que dependerem desta máquina serão afetadas durante este *período de inatividade*, o que igualmente acarretará custos. Deste modo, será preciso a estimativa dos custos por atraso de atividade (discutidos no parágrafo anterior) para *este* período.

Os métodos utilizados para o cálculos destes custos têm-se baseado, principalmente, em métodos "míopes" e métodos do tipo OPT. Estes últimos baseiam-se em recursos, identificando o mais carregado (em algum sentido) e determinando prioridades para o resto do sistema no sentido de minimizar seu uso (por exemplo, uma tarefa que use pouco o recurso gargalo mas requeira processamento intensivo em outras partes do sistema pode ser executada em primeiro lugar). O OPT, na realidade, é um sistema proprietário, da empresa *Creative Output* e, apesar de simplificar o problema de programação focando-o em apenas um recurso crítico, ele não prevê

outras situações importantes, como múltiplos gargalos e gargalos alternantes, bem como exige uma rodada completa do algoritmo quando mudanças reativas são efetuadas.

Como mencionado anteriormente, as estimativas de custos podem ser utilizadas para que diversas decisões sejam tomadas. Em termos de *escolha da seqüência*, por exemplo, os custos podem ser utilizados para calcular as relações de custo/benefício para atribuição de tarefas pelas máquinas, sendo tais tarefas priorizadas em ordem decrescente, entre outras. Para o *job release* (envio da tarefa para o chão-de-fábrica), roteamento, determinação do tamanho de lote (*lot sizing*), preemptividade simples e ociosidade reforçada, outros parâmetros são levados em conta.

2.6 Células de produção e seu emprego nos Sistemas Flexíveis de Manufatura

O conceito de células de produção é baseado no princípio de trabalho em grupo, no qual um pequeno número de pessoas age como uma equipe unida e coesa, visando o desempenho de uma tarefa comum. (Jackson, 1978).

Uma definição "mais prática" é dada por Hyde em Kinney Jr. (1987 – b):

"Células de manufatura constituem uma técnica de produzir lotes médios e pequenos de peças de diferentes tamanhos, formatos e materiais que, entretanto, requerem processos de fabricação similares, desempenhados em pequenos arranjos de máquinas agrupadas, especificamente preparadas e seqüenciadas como uma unidade."

Esta definição leva à identificação dos três passos envolvidos no desenho de uma célula de manufatura:

- Selecionar as peças (ou seja, fazer famílias);
- Selecionar as máquinas;
- Arranjar a célula e designar operadores para ela.

É claro que repensar a produção de uma empresa em termos de células pode levar a uma grande mudança em sua organização, principalmente no que diz respeito aos *lay-outs* de chão-de-fábrica e as manipulações de materiais (*material handling*); em rigor, sua adoção pode introduzir mudanças mais profundas mesmo em nível administrativo (planejamento de processos) e em nível de criação de peças (*design*). Uma destas mudanças é a adequação da célula criada aos elementos externos à esta, impedindo replicação de dados e fabricação desnecessária de produtos (Figura 6):

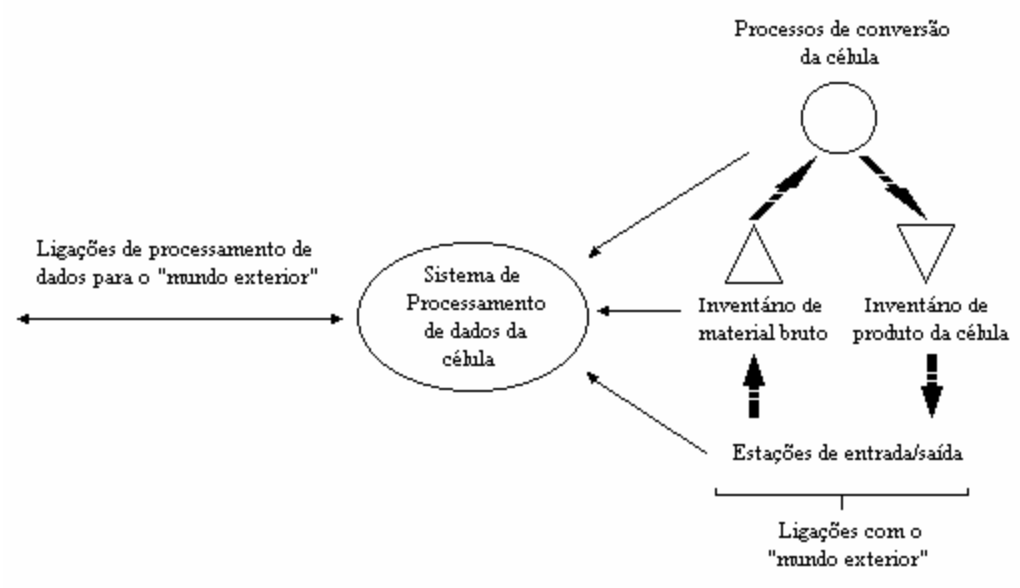


Figura 6: O conceito de célula de um SFM com suas ligações de processamento de dados e manipulação de materiais ao "mundo exterior". (Ránky, 1983)

Kinney Jr. (1987 – a) propõe uma metodologia para o desenho de células de manufatura, que incluem cinco tipos de decisão, dependentes entre si:

- Seleção de peças;
- Seleção de máquinas;
- *Lay-out*, equipamentos e elenco de operadores;
- Avaliação e
- Seqüenciamento e controle da célula.

A seleção de peças determina o conjunto de peças a serem processadas na célula. Naturalmente, estas devem ser similares quanto ao modo como são produzidas. Não apenas esse fator influencia, mas também o maquinário envolvido nessa atividade deve ser, em algum sentido, "semelhante".

Assim, torna-se necessária a consulta a um banco de dados que conterão informações como rotas de processamento, atributos de peças, requerimentos de ferramentas, etc., das quais as técnicas adequadas farão uso para determinar o agrupamento mais adequado.

Estas técnicas constituem objeto de estudo e, entre elas, pode-se citar a *Tecnologia de Grupo* (TG) e a *Análise de Cluster*. Esta última é um procedimento matemático mais complexo, que requer o estudo das distâncias entre máquinas e células das quais fazem parte.

A TG, por sua vez, procura atingir seu objetivo – a economia e a otimização de processos – através de três princípios básicos (Burbidge, 1975):

- *Lay-out* de grupo;
- Ciclo de controle de fluxo curto e
- Seqüência planejada de uso de máquinas.

Assim, a seleção de peças em conjunto com a TG pode facilitar grandemente o projeto de células, sua localização na planta e a interação destas entre si de acordo com as *famílias* de peças (conjunto de partes semelhantes) que produzem.

A *seleção de máquinas*, por seu turno, vai determinar exatamente que máquinas estarão em cada célula bem como a quantidade e o tipo delas. Alguns critérios podem ser levados em conta para esse agrupamento, como nível de utilização, tempo de vida útil do equipamento ou capacidade de gerar gargalos; nesse caso, será desejável evitar o quanto possível que muitas máquinas "críticas" coexistam na mesma célula. Deixa-se claro, porém, que não existem critérios analíticos exatos para este estudo de alocação.

Os *equipamentos*, *lay-out* e *elenco de operadores* também exercem papel de importância, se for considerado que o volume o formato do material está diretamente ligado à manipulação deste pela planta. Também, a quantidade de equipamentos terá influência sobre aquela manipulação quando do projeto da célula. É importante que o *lay-out* seja tal que permita modificações sempre que necessário. Por fim, é desejável que haja operadores *multifuncionais*, ou seja, capacitados a desempenhar todas as atividades na célula.

A *avaliação* diz respeito ao estudo de dimensionamento da célula de acordo com as taxas de produção de peças planejadas.

Por fim, o *controle* e *programação* da célula devem ser igualmente considerados. Como já mencionado, cada célula deve ser encarada como uma unidade dentro da planta e seqüenciada como tal.

Capítulo 3

Formalização teórica

3.1 Introdução aos conceitos principais

A programação de atividades em células de manufatura geral pode ser enquadrada numa categoria particular de problemas, denominada *recurso simples / processador múltiplo* (RSPM). Torna-se importante, portanto, determinar o que é *recurso* e o que é *processador.*

Um *recurso simples* é um grupo de capacidades produtivas que dispõe de uma única fila de entrada e *não possui* nenhum tipo de decisão interna a ser tomada. Se, por exemplo, a disposição das ferramentas no magazine de um centro de fresamento influenciar diretamente nos tempos de *setup* das operações programadas, ocorreu um fato que depende de uma decisão interna: *como* dispor estas ferramentas de modo a minimizar o tempo de *setup* ? Isto caracteriza um *recurso múltiplo*, que não será discutido neste texto. Num recurso simples, as únicas decisões tomadas determinam a seqüência de operações e o tempo de entrada destas no sistema, para processamento (Morton, 1993).

Um *processador simples* (ou máquina), por seu turno, é uma capacidade produtiva programada para lidar com apenas uma atividade ou tarefa por vez *e/ou* com um único tipo de atividade. A máquina capaz de desempenhar diversas atividade simultâneas ou mais de um tipo de atividade é um *processador múltiplo* (Morton, 1993).

Um problema do tipo *recurso simples / processador simples* (RSPS) é, deste modo, o bloco básico utilizado na decomposição de problemas mais complexos. No RSPS, as atividades chegam e deixam o recurso (após processadas) através do tempo, o que o caracteriza como um problema *dinâmico*, muito difícil de resolver. Costuma-se, assim, discutir primeiro as características do problema relaxado nesta condição, denominado RSPS *estático* (no qual se consideram todas as atividades disponíveis no início do horizonte de tempo), e que proverão as condições facilitadoras necessárias ao estudo do problema dinâmico.

A célula de manufatura da FEM – UNICAMP será tratada de um modo ligeiramente diferente da célula padrão. Embora um torno CNC possa desempenhar algumas funções de um centro de usinagem, as três máquinas que a compõem serão empregadas expressamente para suas finalidades principais. Deste modo, pode-se modelar esta célula como um problema do tipo RSPS, e é em função deste modelo que serão estabelecidos todos os resultados discutidos no presente texto.

3.2 Fundamentos básicos do problema RSPS

O problema RSPS é definido para um intervalo de tempo entre um dado início e um dado fim. Cada uma das atividades para este problema possui tempos de chegada ao sistema conhecidos, tempos de processamento e datas limite de entrega. As máquinas podem ter um tempo de *setup* e algumas das atividades podem estar disponíveis para processamento imediato, fornecendo uma fila inicial.

A escolha de um *horizonte de tempo* (período disponível para a execução de todas as atividades) é um assunto delicado: horizontes pequenos exigem menos trabalho computacional, podem tornar a alocação de atividades complicada, ignorando, por exemplo, aquelas que chegam durante a execução da programação e exigem deslocamento das demais, o que provocará penalidades, pois certamente as já programadas excederão seus prazos de entrega. Por outro lado, horizontes maiores exigem mais cálculos e não correspondem, muitas vezes, à realidade, pois a competitividade atual das empresas exigem soluções rápidas e prazos de entrega curtos.

O problema RSPS dinâmico pode ter esta condição relaxada, como já mencionado, tornando-se estático. Nesta modalidade, o RSPS move a chegada de todas as atividades no início do horizonte de tempo (o que pode ocorrer, ocasionalmente, mesmo no problema dinâmico). Embora esta simplificação altere demais o problema, ela fornece bons indícios para a proposição de métodos de resolução para o problema dinâmico.

Outra condição que pode ser relaxada é a introdução da chamada *preemptividade* (ou interrupção), que permite às operações serem interrompidas uma ou mais vezes, para que outras atividades de maior importância possam ser processadas. De modo geral, a preemptividade também propicia facilidades ao estudo do problema dinâmico, embora possa gerar custos adicionais (e neste caso, deseja-se eliminá-los ou, ao menos, minimizá-los). Tais custos poderiam, por exemplo, representar a perda de uma determinada operação pelo fato de não poder ser retomada após a conclusão da operação intercalada pela interrupção. Para efeitos de estudo, entretanto, esta simplificação é útil.

A escolha da função objetivo também é importante pois as heurísticas a serem desenvolvidas levam em conta o tipo de função a ser otimizada. De um modo geral, estas podem ser classificadas em *regulares* e *não-regulares*. Para as primeiras, é sempre preferível concluir as atividades o mais cedo possível, ao passo que para as segundas esta preferência pode não ser conveniente (por exemplo, no caso de ambientes JIT, nos quais terminar atividades muito cedo pode representar excesso de WIP). As funções objetivo regulares podem ser classificadas em termos das *medidas de desempenho* que se quer otimizar:

- Maximizar alguma medida de utilização de recursos;
- Minimizar alguma medida de fluxo de atividade;
- Minimizar alguma medida de adiamento de atividades.

Pode-se, introduzidas as observações acima, classificar os problemas RSPS em:

- Estáticos ou dinâmicos;
- Preemptivos ou não-preemptivos;

- Regulares ou não regulares;
- Com funções objetivo de utilização *x* tempo de fluxo *x* adiamento.

3.3 Definição formal do problema RSPS (Morton, 1993)

A formalização do problema RSPS exige que sejam conhecidas todas as entradas do mesmo e que se faça a distinção entre estas informações e as que são geradas pelas decisões da programação. Deste modo, pode-se estabelecer as seguintes suposições:

- As máquinas processam uma atividade por vez, serialmente;
- O recurso está disponível no horizonte de tempo, entre t_i (início) e t_f (fim) (pode-se assumir $t_i = 0$, para o caso estático);
- *n* atividades simples chegam ao longo do tempo, dentro do horizonte (todas chegando cedo o suficiente para serem processadas);
- Entrada do sistema: tempo de processamento p_j;
- Entrada do sistema: tempo de disponibilidade r_j (= 0 para o caso estático);
- Entrada do sistema: data limite d_j.
- Preemptividade (interrupção): se permitida, possibilita a interrupção de atividades para que outras sejam processadas. Caso contrário, as atividades são desempenhadas sem interrupção;
- Função objetivo: escolha entre regular (permite que todas as atividades terminem o mais cedo possível) e não-regular (todos os outros tipos de função);
- As máquinas estão disponíveis todo o tempo do horizonte e deve concluir todas as atividades ao final deste;
- As entradas são estocásticas e não dependem da seqüência de processamento;
- As datas limites, quando violadas, impõem penalidades à função objetivo.

3.4 Mecânica da programação / medidas de desempenho e funções objetivo

O problema RSPS dinâmico geral tem seus resultados montados seguindo-se os passos:

- Escolhe-se o tempo de retirar a atividade sendo processada da máquina;
- Escolhe-se a próxima atividade a ser processada;
- Escolhe-se o tempo em que a atividade iniciará seu processamento (escolha do tempo de início r_j);
- Repetem-se os três passos anteriores até que não haja mais atividades a serem programadas.

Estes passos produzem uma programação à frente no tempo. Salienta-se que certas restrições devem ser satisfeitas:

- Não se pode processar uma atividade antes que ela chegue na planta;
- Após seu término, uma atividade não pode ser novamente processada;
- Duas atividades não podem ser processadas ao mesmo tempo.

Se a condição relaxada que torna o problema estático for considerada, então o primeiro item anterior pode ser descartado, pois todas estarão já na planta para serem processadas. Do mesmo modo, o primeiro passo da programação pode ser eliminado se for estabelecido que o problema é não-preemptivo: a remoção da atividade j do recurso será sempre p_j unidades de tempo após seu início.

A medida de quão bom é o resultado de uma programação se dá em termos de certas medidas de desempenho, determinadas em função das variáveis discriminadas na formalização do problema RSPS dinâmico (seção 3.3: *Definição formal do problema RSPS*). As principais medidas são (Morton, 1993, Johnson, 1974):

- Tempo de completamento C_j – tempo no qual a atividade j termina seu processamento;

- Tempo de fluxo F_j – é o tempo que a atividade j dispende no sistema, dado por:

$$F_j = C_j - r_j \qquad [1]$$

F_j mede a resposta do sistema à demandas de serviço individuais. O fluxo é a quantidade de tempo entre a chegada da atividade no sistema e sua saída, o que está intimamente relacionado com o custo de WIP. *Lead time do consumidor* e *tempo de resposta do consumidor* são outros nomes para o fluxo.

- Atraso L_j – quantidade de tempo (positiva ou negativa) em que foi excedida a data limite da atividade j:

$$L_j = C_j - d_j \qquad [2]$$

L_j mede a conformidade da programação em relação a uma dada data limite. L_j "recompensa" atividades por se iniciarem cedo, assim como "pune" atividades por se iniciarem mais tarde.

- Adiamento T_j – igual ao atraso da atividade j, se for positivo; se o atraso da atividade j não for positivo, o adiamento é zero:

$$T_j = \max \{0, L_j\} \qquad [3]$$

O adiamento T_j indica que penalidades podem ser imputadas ao atraso de atividades, mas nenhum benefício será tirado caso elas sejam terminadas mais cedo. Existem diversas funções objetivo que medem a quantidade e a frequência dos adiamentos.

- Adiantamento E_j – igual ao negativo do atraso da atividade j, se o atraso for negativo; se o atraso é positivo, o adiantamento é zero:

$$E_j = \max \{0, -L_j\} \qquad [4]$$

Contrário ao adiamento, o adiantamento E_j penaliza atividades encerradas antes do tempo, seja porque o cliente não aceita pedidos antes do prazo, seja porque os custos com estoque sejam maiores do que os custos com o WIP.

A programação de atividades é avaliada, como se sabe, pelas funções objetivo. Estas agregam quantidades que sumarizam o desempenho daquela programação para todas as atividades e podem ser classificadas em três tipos:

- Aquelas que empregam uma média ponderada de uma certa medida de desempenho;
- O máximo ou o mínimo de alguma das medidas;
- Uma composição das duas classificações anteriores.

O peso w_j associado à importância de uma determinada atividade j é o mesmo peso que constará na função objetivo escolhida.

Tendo sido definidas as principais medidas de desempenho da programação, podem ser também definidas as funções objetivo que serão alvo de estudo deste capítulo e empregadas nos testes que serão realizados no dois capítulos seguintes, com o auxílio do *software* SIPMA (Morton, 1993):

- *Makespan*:

$$C_{max} = \max_j \{C_j\} \qquad [5]$$

Esta função objetivo costuma indicar uma alta utilização da planta, pois terminar as atividades programadas mais cedo pode possibilitar o processamento de novas atividades. Esta é uma boa aproximação no sentido de fornecer idéias para problemas mais gerais, mas falha em ambientes do tipo *recurso múltiplo / processador múltiplo* (RMPM), pois uma nova configuração das decisões internas pode gerar um conjunto de atividades mais compacto no tempo.

- Tempo de fluxo ponderado:

$$F_{wt} = \sum_j w_j F_j \qquad [6]$$

- Atraso ponderado:

$$L_{wt} = \sum_j w_j L_j \qquad [7]$$

O *tempo de fluxo* e o *atraso* ponderados são funções populares, de fácil uso e que produzem as mesmas soluções (como será demonstrado no seção 3.8: *Proposições fundamentais*). Além disso, resultados obtidos por estas funções são semelhantes para objetivos ligeiramente diferentes.

- Adiamento ponderado:

$$T_{wt} = \sum_j w_{T_j} T_j \qquad [8]$$

- Número ponderado de atividades adiadas:

$$N_{wt} = \sum_j w_{N_j} \delta(T_j) \text{, em que } \begin{cases} \delta(x) = 1, \text{ se x} > 0 \\ \delta(x) = 0, \text{caso contrário} \end{cases} \qquad [9]$$

- Adiantamento ponderado + adiamento ponderado:

$$ET_{wt} = \sum_j (w_{E_j} E_j + w_{T_j} T_j) \qquad [10]$$

Há ocasiões em que o *adiamento ponderado* (Equação [8]) é um bom objetivo, embora seja difícil de obter soluções exatas para problemas que o empregam. A dinâmica de gargalos (vide seção 2.5.1: *A dinâmica de gargalos*, capítulo 2) pode prover, entretanto, boas heurísticas para tal objetivo, como será visto em itens posteriores. O *adiantamento ponderado + adiamento ponderado* (Equ. [10]) é uma importante função não-regular; é empregada quando os clientes não desejam entregas atrasadas, mas tampouco as aceitarão antes do prazo de entrega. O *número ponderado de atividades adiadas* (Equ. [9]) é um pouco mais rígido: é indicado para clientes que

simplesmente recusam entregas atrasadas. É importante notar que os pesos para fluxo (w_j), adiamento (w_{Tj}), adiantamento (w_{Ej}) e número de atividades adiadas (w_{Nj}) podem ser diferentes.

- Tempo máximo de fluxo:

$$F_{max} = \max_j \{F_j\} \quad [11]$$

- Tempo máximo de atraso:

$$L_{max} = \max_j \{L_j\} \quad [12]$$

- Tempo máximo de adiamento:

$$T_{max} = \max_j \{T_j\} \quad [13]$$

Minimizar o tempo máximo de adiamento é importante quando os clientes suportam pequenos adiamentos na entrega, mas tornam-se rápida e progressivamente mais intolerantes quando estes se alongam. Por outro lado, a minimização do tempo máximo de atraso é útil porque além de ser um problema simples, pode ser usado para a resolução de outros problemas. De fato, se as datas limites das atividades forem igualadas aos tempos de início r_j, minimizar o tempo máximo de atraso também minimiza o fluxo máximo. Do mesmo modo, minimizar o atraso máximo e truncando o valor para 0 caso o valor da função seja negativo minimiza o tempo de adiamento máximo.

Antes de fazer considerações sobre as proposições fundamentais sobre o problema RSPS, é necessário fazer certas observações sobre os pesos citados nas funções. Costuma-se empregar $w_j=1/n$ como uma aproximação que compromete a correta interpretação econômica de tais variáveis. Geralmente, este valor é adotado por falta de estimativas precisas sobre a real importância de uma dada atividade, e como o atraso ou adiantamento desta pode influenciar no valor da função objetivo. Para os testes que serão executados serão usados valores fixos para os

pesos, de modo a uniformizar os resultados. O capítulo 5 (*Resultados e Discussões*) introduzirá tais variáveis de maneira mais detalhada.

3.5 Preemptividade

A interrupção de atividades, se permitida, admite uma aproximação em relação ao tempo contínuo: sua subdivisão em pequenas unidades, que podem ser tornadas tão pequenas quanto se deseje, para recuperar a preemptividade original. Na realidade, este é o conceito de *slot*, já discutido no capítulo 2, seção 2.4 (*Metodologias de modelagem de problemas de programação*). Utilizar a preemptividade pode ser necessário em um determinado tipo de problema, e sua versão *unitária* é muito útil em demonstrações.

A preemptividade unitária introduz o problema de se avaliar, para cada *slot* (e para o segmento de atividade aí contida), o valor da função objetivo. É claro que se uma atividade é desempenhada sem interrupção, a função objetivo sofrerá uma mudança. Caso seja possível de ser interrompida (deste modo permitindo que seja dividida em mini-atividades, uma em cada *slot*), deve ser determinado o novo impacto na função. Existem três métodos principais (Morton, 1993):

- Custo ao fim (CAF): todas as mini-atividades possuem a data limite da atividade original, sendo que a última a ser processada efetivamente modifica a função objetivo;
- Custo por extensão (CE): como no método anterior, as mini-atividades possuem a data limite da atividade original. O valor da função objetivo para cada uma delas é $1/p_j$ do valor da função para a atividade total;
- Custo por extensão modificado (CEM): o valor da função permanece $1/p_j$ para cada mini-atividade; o que se altera são as datas limites, que passam a ser, seqüencialmente, d_j, $(d_j - 1)$, $(d_j - 2)$,.....

As três atribuições de custo acima e os dois conceitos a serem vistos a seguir são ferramentas importantes nas demonstrações de alguns dos resultados a serem estudados na seção

3.7 (*Proposições fundamentais*) e com mais detalhes no seção 3.9 (O RSPT dinâmico – resultados exatos)

3.6 Dominância, troca par-a-par transitiva e prioridades

A *dominância* (Morton, 1993) é qualquer propriedade que especifique um subconjunto de todas as seqüências de um programa que contenha, garantidamente, uma seqüência ótima. A dominância reduz o número de possíveis soluções que deve ser considerada, embora possa eliminar outros candidatos à seqüência ótima. Quando forem estudadas as heurísticas para o problema RSPS com função objetivo T_{wt} (bem como para outras) serão apresentados exemplos práticos do emprego da dominância.

A troca par-a-par transitiva é uma ferramenta usada para demonstrar que certas regras de decisão fornecem seqüências ótimas para determinados problemas. A idéia é trocar de posição elementos adjacentes da seqüência, até que seja determinado que nenhuma outra troca incrementará (no sentido de melhorar) a função objetivo. As definições a seguir também são dadas por Morton (1993):

> **Definição:** para duas atividades adjacentes *i* antes de *j*, começando no tempo *t*, define-se a *relação de ordem dinâmica iR(t)j* significando que "começando no tempo *t*, *i* programada antes de *j* é melhor que *j* programada antes de *i*". Em outras palavras, trocar *i* e *j* de posição não melhorará a função objetivo.

> **Definição:** A *relação de ordem global (ou estacionária) iRj* significa "iR(t)j sempre que *i* e *j* sejam adjacentes, em qualquer tempo *t*".

Diferente da definição anterior, a relação de ordem global indica que a troca entre *i* e *j* é sempre recomendada, não interessando se *i* e *j* estão adjacentes no começo, no meio ou no fim do horizonte. Para problemas regulares "bem comportados" como T_{wt} e F_{wt} todas as relações serão do tipo global.

Definição: *R* é globalmente transitiva para um problema se a relação de ordem é global para cada par de atividades *iRj* e *jRk* sempre implica *iRk*.

As três definições acima levam a um resultado útil:

Proposição 1: para um problema regular e estático, se a relação de troca *R* é globalmente transitiva, então trocas de posição entre atividades adjacentes garantem a produção de uma solução ótima. A solução pode ser ordenada por *R*.

Demonstração: sejam consideradas quaisquer atividades *i* e *j*, em qualquer tempo *t*. Sejam consideradas também tanto *iR(t)j* como *jR(t)i*. Pela propriedade global, tanto *iRj* como *jRi* são válidos. Deste modo, pela transitividade, os elementos podem ser simplesmente ordenados. Se *k* for o elemento mais dominante da seqüência, ele pode ser movido para o primeiro elemento desta por trocas par-a-par, bem como o mesmo pode ser feito com o segundo elemento mais dominante, e assim sucessivamente.

Como será introduzido no seção 3.8 (*Proposições fundamentais*), uma função objetivo Z pode ser escrita como uma soma de $f_i(t)$ para cada atividade *i*, no qual *t* é o tempo de completamento da atividade *i*. Considerando o tempo de processamento p_j de cada atividade como 1, tem-se a seguinte

Definição: a *prioridade unitária* π de uma atividades *i* é dada pela taxa de crescimento de sua função objetivo por unidade de tempo que seu completamento incrementa. Em outras palavras, $\pi(i, t) = f_i(t+1) - f_i(t)$.

As heurísticas de despacho calculam as prioridades para todas as atividades em $t = 0$ e então programam a atividade de maior prioridade primeiro. Todas elas são recalculadas no tempo p_i (tempo de completamento da atividade programada em primeiro lugar) para avaliar a operação a ser programada em segundo lugar, e assim sucessivamente. Deste modo, o número de prioridades a serem avaliadas é $n + (n - 1) + (n - 2) + ... = n(n + 1)/2$.

Existem prioridades que são independentes do tempo, razão pela qual apenas n prioridades são calculadas nestes casos.

A definição de prioridade permite estabelecer uma função que indique o efeito na função objetivo da troca de posição entre a atividade i e a atividade j.

Proposição 2: o efeito da troca unitária (fazendo $p_k = 1$) entre duas atividades i e j é a diferença entre suas duas prioridades: $D(i, j, t) = \pi(i, t+1) - \pi(j, t+1)$.

Demonstração: $D(i, j, t) = f_i(t + 2) - f_i(t + 1) + f_j(t + 1) - f_j(t + 2)$. Basta agora aplicar a definição de π.

Proposição 3: para o caso cujos tempos de processamento são p_j, se para algum $\pi(i, t + m_1) \geq \pi(j, t + m_2)$ para todo j e para todo m_1 e m_2 tal que $1 \leq m_1, m_2 \leq 1 + 2\max_k(p_k)$, então a atividade i de maior prioridade vai dominar a troca par-a-par adjacente com qualquer outra atividade no tempo t.

Demonstração: Morton (1993).

A combinação da Proposição 3 com a Proposição 1 fornece resultados fortes, no sentido de que existe transitividade local em um dado ponto no tempo e que a transitividade global garante uma troca entre atividades adjacentes que produza uma seqüência ótima. Entretanto, a utilização destes resultados requer que as mudanças de prioridades sejam suaves, a cada iteração. Além disso, a transitividade local não implica em transitividade global, motivo pelo qual deve-se dispor de uma boa heurística, para que as trocas transitivas locais permaneçam, em algum sentido, estáveis, e possam, deste modo, ter maiores chances de conduzir a uma solução ótima.

3.7 Proposições fundamentais

Para que algumas regras de decisão para o problema estático RSPS possam ser estabelecidas, é necessário que algumas definições sejam introduzidas.

Definição: uma *função objetivo baseada em completamento* é denotada por $Z = f(C_1, C_2, ..., C_n)$.

Definição: Z é *aditiva* se $Z = \sum_j f_j(C_j)$.

Definição: Z é *maxi* se $Z = \max_j \{f_j(C_j)\}$.

Definição: Z é *regular* se Z cresce apenas se pelo menos algum C_j cresce.

Acerca destas definições, têm-se:

Proposição 4: se Z é aditiva, ela será regular se e somente se cada uma das funções $f_i(C_j)$ é monotonicamente crescente em C_j.

Proposição 5: se Z é maxi, ela será regular se cada função $f_j(C_j)$ é monotonicamente crescente; será não-regular se pelo menos uma função $f_k(C_k)$ é localmente decrescente para pelo menos um conjunto de completamentos (C_1, C_2, ..., C_n) para o qual $f_k(C_k) = \max_j \{f_j(C_j)\}$.

Demonstração das proposições 4 e 5: Morton (1993).

A próxima proposição estabelece um resultado para problemas regulares e dinâmicos, sem preemptividade.

Proposição 6: para o problema RSPS dinâmico, sem preemptividade, a programação é unicamente definida pela seqüência de atividades. A próxima atividade na seqüência se inicia assim que chega e a máquina correspondente torna-se livre, com:

$$C_j = \max\{C_{j-1}, r_j\} + p_j \qquad [14]$$

Ou

$$C_j = C_{j-1} + \max\{(r_j - C_{j-1}), 0\} + p_j \qquad [15]$$

Demonstração: apesar de a seqüência provavelmente forçar algum tempo ocioso enquanto a próxima atividade chega, a regularidade assegura que a próxima atividade se iniciará assim que possível. O termo do meio da segunda expressão para C_j é a ociosidade inserida.

Proposição 7: para o mesmo caso anterior, considerando agora que a ociosidade inserida não é permitida (ou seja, para toda atividade na seqüência tem-se $r_j \leq C_{j-1}$), as relações deduzidas na Proposição 4 reduzem-se a:

$$C_j = \sum_{i=1}^{j} p_i \qquad [16]$$

Um caso especial importante é $r_j = 0$, ou seja, o caso estático.

Demonstração: basta fazer todos os termos em r = 0, uma vez que a ociosidade inserida não é permitida.

3.8 Casos exatos para o problema RSPS regular estático

As idéias discutidas a seguir fornecem regras exatas intuitivas para algumas das funções objetivo listadas na seção 3.4 (*Mecânica da programação / medidas de desempenho e funções objetivo*). Antes, porém, é necessário classificar as funções em três categorias:

- Funções baseadas em utilização;
- Funções baseadas em fluxo e
- Funções baseadas em datas limite.

3.8.1 Funções baseadas em utilização

Todas as funções baseadas em utilização são aplicadas em problemas nos quais a planta é intensamente utilizada. Embora o *makespan* seja um indicador limitado, seu emprego no problema RSPS estático é particularmente fácil de analisar, o que leva à seguinte

Proposição 8: qualquer seqüenciamento de atividades que não envolva ociosidades leva a um *makespan* mínimo, que é $C_{max} = \sum_{j=1}^{n} p_j$.

Demonstração: para qualquer problema RSPS, estático ou dinâmico, o *makespan* é a soma de todos os tempos de processamento mais a soma dos tempos ociosos. Deste modo, qualquer seqüência sem ociosidade inserida é ótima para o problema estático.

As limitações do *makespan* repousam no fato de que esta medida facilmente perde seu valor em casos mais difíceis. Por exemplo, quando o problema considera tempos de *setup* dependentes de seqüência ou é do tipo RMPM, a estrutura usando o *makespan* torna-se extremamente complexa. Outro detalhe a ser observado é que o horizonte de tempo seja escolhido de tal maneira a não ocorrer nenhuma máquina ocupada em seu final, ou perto dele. Caso isto não aconteça, é preciso corrigir o intervalo de tempo, o que não é um procedimento trivial.

3.8.2 Funções baseadas em fluxo

Quando o mais importante numa planta são os *lead times* de entrega (ou a redução do WIP ou ainda em ambientes JIT), a redução do fluxo de atividades é um fator de peso a ser considerado na função objetivo.

Proposição 9: para o problema com função objetivo F_{wt}, se todos os $p_j = 1.0$, então uma seqüência ótima deve satisfazer $w_1 \geq w_2 \geq .... \geq w_n$.

Demonstração: seja uma seqüência ótima com $w_j < w_{j+1}$. Trocar as posições entre w_j e w_{j+1} gerará um fluxo ponderado decrescido de $w_{j+1} - w_j$. Como esta nova seqüência é melhor do que a anterior, a seqüência original não era ótima.

A próxima proposição fará uso da anterior e do método de preemptividade *custo por extensão* (CE) para derivar uma regra geral para problemas do tipo F_{wt}:

Proposição 10: para o problema com função objetivo F_{wt}, uma seqüência ótima de atividades deve satisfazer $(w_1/p_1) \geq (w_2/p_2) \geq ... \geq (w_n/p_n)$.

Demonstração: seja formada a preemptividade CE, na qual as mini-atividades possuem impacto w_j/p_j na função objetivo. Usa-se agora a proposição 9 para resolver este problema relaxado, que não terá sua solução alterada caso sejam rearranjadas as mini-atividades de mesmo peso. Deste modo, pode-se formar uma solução não-preemptiva com o mesmo custo (portanto, que é ótima).

A regra acima é denominada *Menor Tempo Ponderado de Processamento* (MTPP ou *Weighted Shortest Processing Time* – WSPT).

Proposição 11: para o problema RSPS estático ou dinâmico, qualquer um dos seguintes objetivos de minimizar:

- Fluxo ponderado;
- Tempo de completamento ponderado;
- Atraso ponderado e
- WIP ponderado.

possui uma seqüência ótima dada por MTPP.

Demonstração: sejam as três primeiras funções. O fluxo ponderado é

$$F_{wt} = \sum_j w_j F_j = \sum_j w_j (C_j - r_j) = \sum_j w_j C_j - \sum_j w_j r_j = C_{wt} - r_{wt} \qquad [17]$$

O tempo de completamento ponderado é:

$$C_{wt} = \sum_j w_j C_j = C_{wt} \qquad [18]$$

O atraso ponderado é:

$$L_{wt} = \sum_j w_j L_j = \sum_j w_j (C_j - d_j) = \sum_j w_j C_j - \sum_j w_j d_j = C_{wt} - d_{wt} \qquad [19]$$

Note-se que r_{wt} e d_{wt} são constantes (combinações ponderadas de tempos de chegada e datas limite) e não influenciam na solução. Deste modo, os valores das funções objetivo diferem apenas por uma constante, motivo pelo qual podem ser minimizadas pela mesma regra de decisão MTPP. Como mencionado anteriormente, os pesos são variáveis que indicam componentes econômicos das atividades. Assim sendo, pode-se interpretá-los como, por exemplo, custo diário de inventário, o que automaticamente classifica o WIP ponderado como uma medida de fluxo, tornando-o possível de ser minimizado via MTPP.

3.8.3 Funções baseadas em datas limite

Os objetivos baseados em datas limite são importantes quando se deseja minimizar a frequência com que as atividades as excedem, satisfazendo assim os clientes com tempos de entrega confiáveis. Atraso máximo, número ponderado de atividades adiadas e adiamento máximo são algumas das funções desta classe e são relativamente simples.

Proposição 12: para o problema estático de minimizar o atraso máximo, a solução ótima satisfaz $d_1 \leq d_2 \ldots \leq d_n$. Esta é a *regra da Data Limite Mais Adiantada* (DLMA) ou *Earliest Due Date* (EDD).

Demonstração: seja primeiramente considerado que $p_k = 1.0$, para todo $k = 1,...n$. Seja considerado agora que existem duas atividades tais que $d_j > d_{j+1}$. A atividade $j+1$ tem uma data limite pelo menos uma unidade mais adiantada do que j e é processada uma unidade de tempo mais tarde. Deste modo, o atraso de $j+1$ é pelo menos duas unidades em relação a j, razão pela qual se houver uma troca adjancente o atraso de $j+1$ cairá uma unidade, compensando em j, diminuindo o atraso máximo. Considerando agora o problema original, basta formar a preemptividade CE e, do mesmo modo que a proposição 10, pode-se rearranjar as mini-atividades

de mesma data limite até ser formada uma solução não-preemptiva, sem custo adicional: portanto, é ótima.

A política DLMA é boa para atrasos relativamente grandes, não tendo a mesma eficiência para atrasos menores.

Proposição 13 (Minimizar o adiamento máximo): a minimização do adiamento máximo se dá, como o atraso máximo, pela política DLMA.

Demonstração: se a minimização do atraso máximo gera um valor positivo, o adiamento máximo não pode gerar um valor menor. Se o atraso máximo gera um valor negativo, o adiamento zero é possível de ser obtido por DLMA. (rever as definições de atraso e adiamento, na seção 3.4 (*Mecânica da programação / medidas de desempenho e funções objetivo*).

Proposição 14 (Número ponderado de atividades adiadas): para o problema estático com todos os $p_j = 1$, os seguintes passos:

- Ordenar as atividades de acordo com a política DLMA;
- Parar, caso não haja atividades adiadas. Caso contrário, seguir;
- Encontrar a primeira atividade adiada na seqüência (índice k, na seqüência);
- Mover a atividade com menor peso w_{Nj}, $1 \leq j \leq k$ para o final da seqüência;
- Recalcular os tempos de completamento e voltar ao passo 2.

garantem a obtenção de uma seqüência ótima.

Demonstração: qualquer subconjunto da seqüência estará em ordem DLMA. Em particular o estará a seqüência que contém as atividade de 1 até k (a primeira atividade adiada). A atividade $k-1$ não está adiada, sendo que k tem tempo de completamento apenas uma unidade maior do que $k-1$, com $d_{k-1} \leq d_k$. Portanto, k estará adiada por apenas uma unidade de tempo, e se for removida a atividade com menor peso, k não estará mais adiada. Executando este procedimento repetidamente até que não haja mais atividades adiadas (com exceção das que foram movidas para o final da lista) obtém-se a solução ótima.

É interessante notar que todas as atividade adiadas estarão no final da seqüência, podendo simplesmente ser ignoradas, de acordo com as penalidades imputadas.

3.8.4 O emprego de prioridades e da dinâmica de gargalos

Outra metodologia para a obtenção de heurísticas para problemas tipo RSPS estáticos é considerar a dinâmica de gargalos, que empregará noções de custo (rever capítulo 2, seção 2.5.1: *A dinâmica de gargalos*). De um modo geral os custos envolvidos estão em termos da relação benefício / custo, como se pode notar abaixo:

- Priorizar as atividades em ordem decrescente da relação benefício / custo (esta é a regra de decisão para a programação);
- O benefício de escolher uma atividade é a importâncida da atividade multiplicado por um fator de folga, que mede a "distância" da conclusão da atividade em relação à sua data limite;
- O custo de escolher uma atividade é custo de expedição remanescente do recurso para a tarefa da qual a atividade faz parte;
- Este curso de expedição remanescente é a taxa de expedição multiplicado pelo custo do recurso remanescente;
- O custo do recurso remanescente é o tempo de processamento de cada uma das atividades restantes multiplicado pelo preço do(s) recurso(s) que estas usam, somado para todas elas.

A conjugação destes dois conceitos é muito útil, podendo ser extensivamente estudada em trabalhos futuros. Para o presente texto, serão empregados para derivar heurísticas adicionais para o problema de fluxo ponderado e adiamento ponderado (F_{wt} e T_{wt}) para os problemas RSPS estático e para alguns métodos dinâmicos.

Conforme comentado no seção 2.5.1, a dinâmica de gargalos (DG) estima os valores obtidos ao se atrasar a entrega de uma tarefa, ou o uso de um determinado recurso, trabalhando

com estes valores e determinando as prioridades de programação (seqüenciamento). Deve-se, portanto, dispor de um método de atribuir custos a estas duas variáveis.

Se *R(t)* for o preço por unidade de tempo de se usar o recurso no tempo *t* e *I* é uma taxa derivada do custo de capital da firma, então *IR(t)* é o valor de se usar o recurso antes ou depois do tempo. Seja também considerada a folga da atividade *j*, dada por $S_j(t) = d_j - p_j - t$ e defina-se $U_j(t) = f_j(S_j(t))$ como a urgência de tal atividade, sendo que f_j é o custo marginal de fazer decrescer a folga de *j* na função objetivo. Deste modo, o preço por atrasar a atividade *j* é dado por $w_j U_j(t)$.

Resumindo:

- O custo de se atrasar (ou adiantar) o uso de um dado recurso é $IR(t)$;
- O custo de se atrasar a atividade j é $w_j U_j(t)$.

Seja uma atividade *j* para a qual se deseja determinar a prioridade. O custo por se adiar a programação desta atividade por uma certa quantidade de tempo Δ será $\Delta w_j U_j$. Isso resultará, identicamente, a um custo ΔIRp_j pelo uso adiado do recurso, de modo que o custo líquido final será $\Delta w_j U_j - \Delta IRp_j$. Uma regra de prioridade, portanto, para cada recurso, poderia ser dada por:

$$\pi_j = \Delta\left[\frac{w_j U_j}{IRp_j} - 1\right] \qquad [20]$$

Donde, se for considerado que Δ, I e R são iguais para cada recurso, tem-se que:

$$\pi_j = (w_j / p_j) U_j \qquad [21]$$

U_j variando com o tempo é um detalhe importante a ser levado em conta; porém, em certos casos, isto não acontece, como por exemplo, no caso de se efetuar a minimização do atraso máximo. Particularmente, para esta função objetivo, tem-se que:

$$\pi_j = w_j / p_j \qquad [22]$$

Pois U_j é invariante (logo, uma constante). Pode-se notar que esta regra coincide com a proposição 10, demonstrada no seção 3.9.2.

Uma regra de prioridade invariante (Morton, 1993) com o tempo para minimizar o adiamento ponderado leva o nome de seu criador: Montagne. De acordo com esta regra, a prioridade para seqüenciar as atividades pode ser escrita como:

$$\pi_j = \left(w_j / p_j \right) \left[1 - \left(d_j / \sum_i p_i \right) \right] \qquad [23]$$

Casos que envolvem a função de minimizar o adiamento ponderado dependente do tempo utilizam efetivamente a função $S_j(t) = d_j - p_j - t$. Uma das heurísticas propostas para casos como este é proposta por Carroll (1965):

$$\pi_j = \left(w_j / p_j \right) \left[1 - \left(S_j / k p_j \right) \right], \text{ para } S_j > 0 \text{ e } (1 - [S_j / k p_j]) > 0 \qquad [24]$$

k é um parâmetro real livre.

Pode-se notar que as duas regras obtidas para T_{wt} são do tipo MTPP, multiplicado por um fator de folga.

3.9 O RSPS dinâmico – resultados exatos

Será tratado agora o problema RSPS cujas chegadas variam com o tempo (ou seja, nem todos os r_j são iguais a zero). Este é, obviamente, um problema mais comum e mais natural no sentido de a célula de manufatura não estar isolada, mas inserida num contexto maior, recebendo pedidos de outros pontos da fábrica, em tempos distintos.

O fato de estarem sendo consideradas chegadas dinâmicas ao sistema exige que a *ociosidade inserida* agora seja levada em conta. Além disto, a preemptividade CAF e CE serão mais empregadas para derivar resultados exatos.

Existe, para o modelo dinâmico, uma *fila* de atividades, que pode estar vazia (todas as atividades estão sendo processadas na planta e não há nenhuma esperando para iniciar seu processamento). Uma *fila de sistema*, por sua vez, engloba a fila e as atividades que estão sendo processadas no momento.

Os resultados apresentados nesta seção possuem demonstrações semelhantes aos seus análogos no caso estático, não justificando uma nova demonstração. Deste modo, afirma-se que as regras definidas para os casos dinâmicos produzem seqüências ótimas para as funções objetivo correspondentes.

Pode-se definir, para os casos preemptivos dinâmicos de minimizar F_{wt} ou L_{wt}, a seguinte regra (*Menor Tempo Ponderado de Processamento Preemptivo Dinâmico – MTPPPD*):

- Quando uma atividade se completa, iniciar a próxima com o maior w_j/p_j (pela disciplina de preemptividade CE);
- Quando uma atividade i chega ao sistema, comece a processá-la caso $w_i/p_i > w_j/p_j$; caso contrário, continua-se com o processamento de j (disciplina CE);
- Caso a disciplina de preemptividade seja por CAF, deve-se substituir pj nos dois passos anteriores por p_{rj}, que é o tempo remanescente de processamento de j.

Percebe-se que a máquina deve estar sempre designada para as atividades de maior relação peso / unidade de tempo de processamento *remanescente* (CAF) ou *total* (CE).

Para os problemas dinâmicos de minimizar o atraso máximo (L_{max}) ou o adiamento máximo (T_{max}), basta considerar que a disciplina DLMA fornece uma seqüência ótima (rever seção 3.8.3: *Funções baseadas em datas limite*) e assim definir a disciplina DLMAPD (Data Limite Mais Adiantada Preemptiva Dinâmica ou PEDD – *Preemptive Dynamic Earliest Due Date*):

- Programar a atividade com menor data limite;
- Se uma nova atividade chegar à planta, com data limite menor, interromper a atividade já sendo processada para que esta o seja.

É interessante notar que nenhuma previsão ou qualquer outro método de determinação das chegadas são necessários, mantendo o caráter de despacho destas aproximações.

Os resultados para o número ponderado de atividades adiadas são parciais e não serão discutidos. Porém, para o adiamento ponderado, afirma-se que se todas as atividades possuírem a mesma data limite, então a seqüência ótima será dada por MTPPPD.

3.9.1 Ociosidade inserida não permitida

Existem diversas semelhanças entre o problema RSPS estático e dinâmico sem ociosidade inserida permitida. Entretanto, não existem resultados exatos para este caso e os métodos disponíveis têm utilidade para uma pequena classe de problemas preemptivos. A regra de "ociosidade inserida não permitida" faz com que nenhum dos recursos da planta fique desabilitado para o tratamento de atividades repentinamente introduzidas para processamento (vindas, por exemplo, de ordens cujo cumprimento é urgente).

Para os casos de minimizar L_{wt}, F_{wt} e L_{max} são utilizados os procedimentos do típo *míope*, baseados apenas nas atividades que já estão disponíveis para serem processadas. Entretanto, por este mesmo motivo, tais procedimentos podem introduzir falhas, cuja correção não é trivial. De qualquer modo, esta aproximação é útil quando se trata da resolução de problemas maiores. Resumidamente, pode-se escrever:

- Seja dado um certo conjunto de atividades não processadas até o tempo t;
- As prioridades para estas atividades são calculadas como no caso estático;
- A atividade com a maior prioridade no conjunto é processada.

Para L_{wt} e F_{wt}, portanto, emprega-se MTPP; para T_{max} e L_{max} pode ser usado DLMA e para T_{wt} pode ser empregada a heurística de Montagne ou a de Carroll (Morton, 1993)

3.9.2 Ociosidade inserida permitida

As plantas dinâmicas nas quais a ociosidade inserida é permitida dão margem ao processamento das atividades urgentes, pois os recursos não estarão alocados exclusivamente para as atividades que já se encontram no sistema. Ao contrário, cada recurso dispõe de um certo tempo ocioso durante o qual pode ser alocado para processar atividades externas. A definição deste tempo ocioso é uma das questões fundamentais neste modelo, assim como deve-se determinar a relação entre a prioridade da atividade que vai utilizar o recurso, já estando na fila e da atividade recém-chegada.

Para objetivos regulares, o tempo ocioso não precisa ser maior do que o menor tempo de uma atividade esperando para ser processada, pois esta pode ocupar precisamente aquele tempo. Assim, a solução pode ser obtida fazendo permutações entre as operações, o que sugere o emprego do conceito de troca adjacente, já estudado na seção 3.6 (*Dominância, troca par-a-par transitiva e prioridades*).

Sejam duas atividades i e j, com tempos de disponibilidade r_i e r_j. Em t = 0, deseja-se avaliar o efeito da troca entre i e j. Assume-se que uma atividade chegará ao recurso depois que a outra já tiver sido processada. Se i for programada primeiro, então

$$C_i = r_i + p_i \text{ e } C_j = r_i + p_i + p_j = C_i + p_j. \qquad [25]$$

Se j for programada antes, então

$$C_j = r_j + p_j \text{ e } C_i = r_j + p_j + p_i \qquad [26]$$

Programar j antes de i faz com que:

- i seja atrasada por $p_j + (r_j - r_i)$ unidades de tempo;
- j seja atrasada por $-p_i + (r_j - r_i)$ unidades de tempo;
- Todas as demais atividades subseqüentes são atrasadas por $(r_j - r_i)$.

Da seção 3.8.4 (*O emprego de prioridades e da dinâmica de gargalos*), tem-se (a menos do fator Δ, uma constante) que:

$$w_j U_j - IRp_j \qquad [27]$$

Que é o custo por unidade de tempo de se processar j mais tarde, assim como para i. Somando-se $p_j + (r_j - r_i)$, $-p_i + (r_j - r_i)$ e $(r_j - r_i)$, dividindo o resultado por IRp_ip_j e sendo (seção 3.8.4):

$$\pi_j = \left[\frac{w_j U_j}{IRp_j} - 1 \right] \qquad [28]$$

Obtém-se:

$$\pi_{j\text{-ocioso}} = \pi_j - r_j \left[\frac{1}{Ip_i p_j} + \frac{p_j}{p_i} + \frac{p_i}{p_j} \right] \qquad [29]$$

A dificuldade da regra de prioridade acima é a sua dependência, para uma atividade, de seu par. Tornando a prioridade de j muito maior que a de i (de modo a poder ser desconsiderada) e considerando o tempo de processamento médio de todas as atividades, têm-se, finalmente:

$$\pi_{j\text{-ocioso}} = \pi_j(1 - (r_j - t)/p_{av}) - ((r_j - t)/p_j)(1/Ip_{av}), \text{ para } (r_j - t) > 0 \qquad [30]$$

Esta regra toma a prioridade da atividade j e a faz decrescer de dois fatores relacionado ao tempo r_j. O segundo fator leva em consideração a taxa I, que no caso do presente trabalho não será utilizada diretamente, por derivar de fatores econômicos desconsiderados por simplificação.

Pode-se, entetanto, ignorar o termo que leva em consideração esta taxa (que envolve os custos de utilização do recurso) por um termo multiplicando $(r_j - t)/p_{av}$ indicando a taxa de utilização efetiva da máquina. Portanto, tem-se que:

$$\pi_{\text{j-ocioso}} = \pi_j(1 - F(r_j - t)/p_{av}), \text{ para } (r_j - t)>0 \qquad [31]$$

Capítulo 4

Análise experimental

4.1 O *software* SIPMA

O *Sistema Integrado de Planejamento de Manufatura e Automação* (SIPMA) é um *software* desenvolvido pelo autor deste trabalho em conjunto com o Departamento de Engenharia de Fabricação (DEF) – Faculdade de Engenharia Mecânica (FEM) – UNICAMP, cujos recursos foram empregados para tal fim. Presta-se a auxiliar o planejamento e seqüenciamento de operações em termos práticos, segundo os métodos e algoritmos anteriormente discutidos.

Por meio da interface entre um ambiente de programação e um banco de dados, é possível manipular as informações devidamente armazenadas e construir tabelas de seqüenciamento (cartas de Gantt) segundo os critérios de cada heurística; em seguida, pode-se visualizar as medidas de desempenho e valores das funções-objetivo definidas no capítulo 3 para comparação e análise.

O desenvolvimento deste sistema foi efetuado utilizando-se a linguagem *Delphi* (da *Borland*®) em sua versão 3, escolhida pela sua capacidade de manipulação de várias fontes de dados, entre as quais o *Microsoft*® *Access* (Microsoft), utilizado intensivamente para armazenamento e recuperação de informações. Por ser uma versão visual do conhecido Pascal, a programação pôde ser beneficiada por todas as estruturas de dados que esta linguagem oferece.

4.1.1 Sobre a operação do *software*

O SIPMA, em sua tela principal, apresenta um menu em que é possível:

- Cadastrar dados relativos à máquinas, peças e operações relacionadas, bem como seus detalhes;
- Escolher a heurística a ser utilizada na resolução do modelo;
- Exibir os resultados da programação via representação visual (Carta de Gantt) ou de arquivo texto, para uso em relatórios e
- Ter acesso às várias medidas de desempenho e funções-objetivo definidas no capítulo anterior (páginas 35 a 39) para que possa ser estabelecida uma comparação entre as heurísticas programadas em termos destes valores.

O correto emprego do sistema compreende três passos de simples execução: o cadastro das informações que comporão o modelo, a geração de um cenário de execução em um dado mês, a resolução do modelo e, por fim, a visualização do resultado obtido. Segue o detalhamento do sistema, seu uso e sua estrutura interna para que, no capítulo 5, resultados de sua aplicação possam ser analisados e discutidos.

4.1.1.1 Cadastro de informações para a execução do modelo

Para que o SIPMA funcione adequadamente, é necessário determinar quais máquinas estão disponíveis e seus custos de operação; as operações e seus detalhes e as peças que serão processadas. Isto é feito por meio do menu *Editar* do sistema (Figura 7):

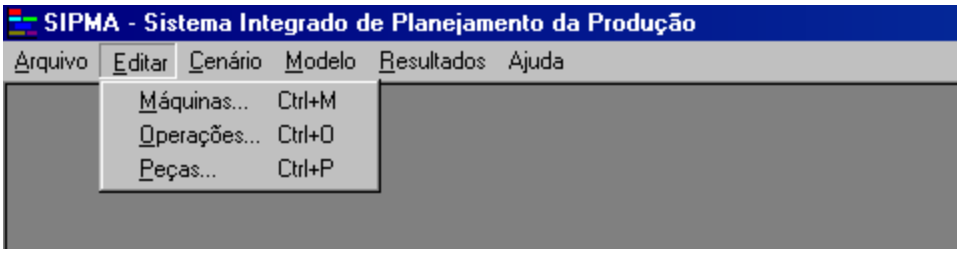


Figura 7: Menu principal do SIPMA – opção "Editar".

O *Cadastro de Máquinas* permite ao usuário identificar as capacidades produtivas da planta, designando-lhe um nome e um custo de operação por unidade de tempo. Os botões presentes na parte superior de todos os formulários de cadastro permitem diversas funções como inserção de um novo registro, edição, salvamento, exclusão, navegação, entre outros (Figura 8).

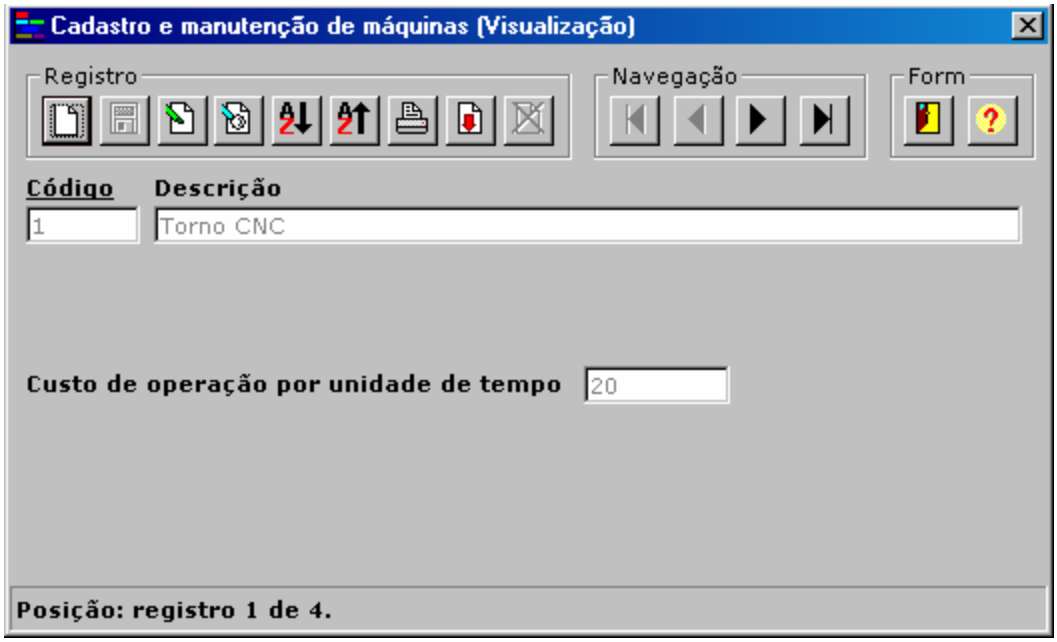


Figura 8: Cadastro e manutenção de máquinas no SIPMA.

O *Cadastro de operações*, por seu turno, é empregado para registrar o elenco de operações que as máquinas podem processar (Figura 9).

Durante o cadastro de uma operação é necessário que seja(m) definida(s) a(s) máquina(s) que a executará(ão). Como se trata de um sistema modelado à luz dos conceitos de FMS, uma operação pode ser desempenhada em mais de uma máquina, quando há disponibilidade da mesma. Esta especificação se dá por meio do botão "*Clique para editar os detalhes desta operação...*", que abre o formulário apropriado (Figura 10). Até 8 máquinas podem ser atribuídas à operação em edição.

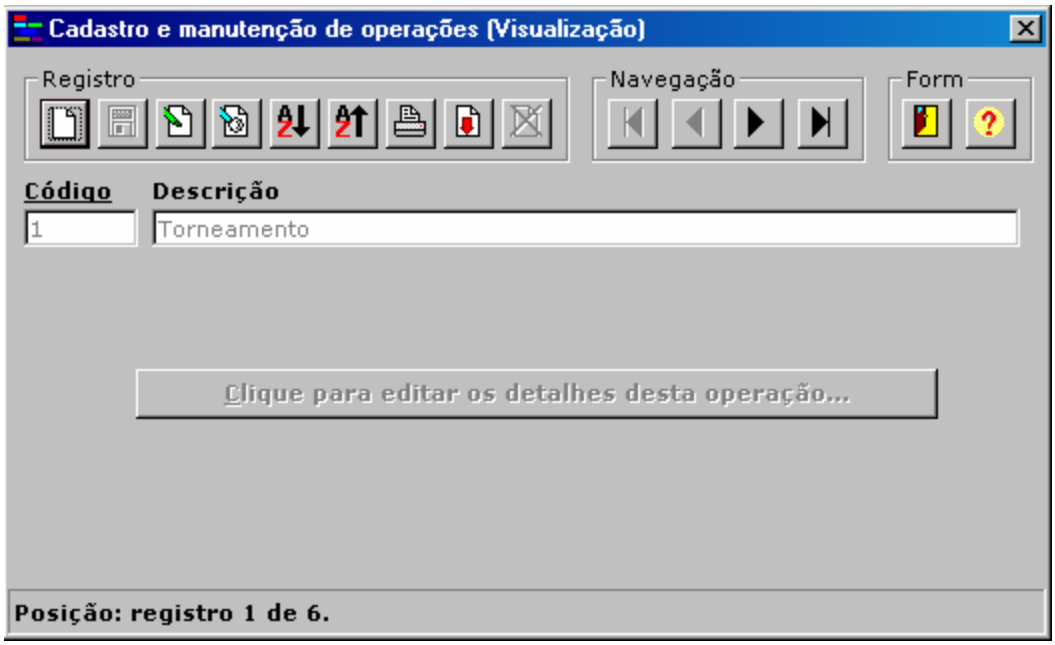


Figura 9: Cadastro e manutenção de operações no SIPMA.

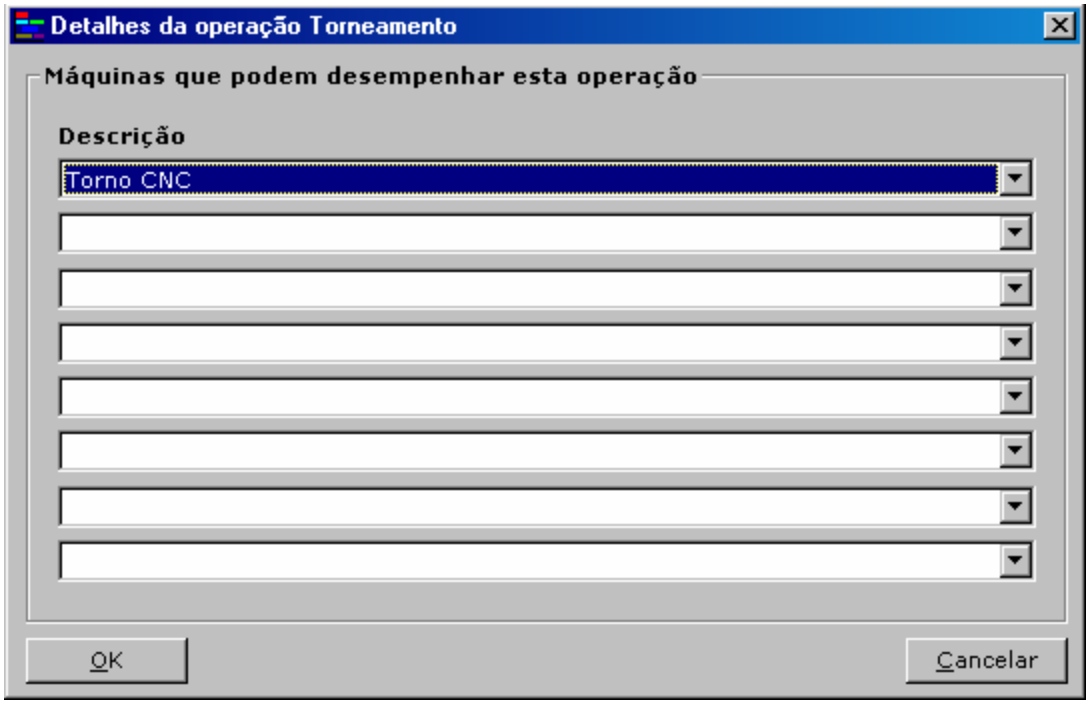


Figura 10: Detalhamento das máquinas que podem desempenhar a operação em edição.

O *Cadastro de Peças*, por fim, permite que sejam registradas as peças que a planta produzirá (Figura 11):

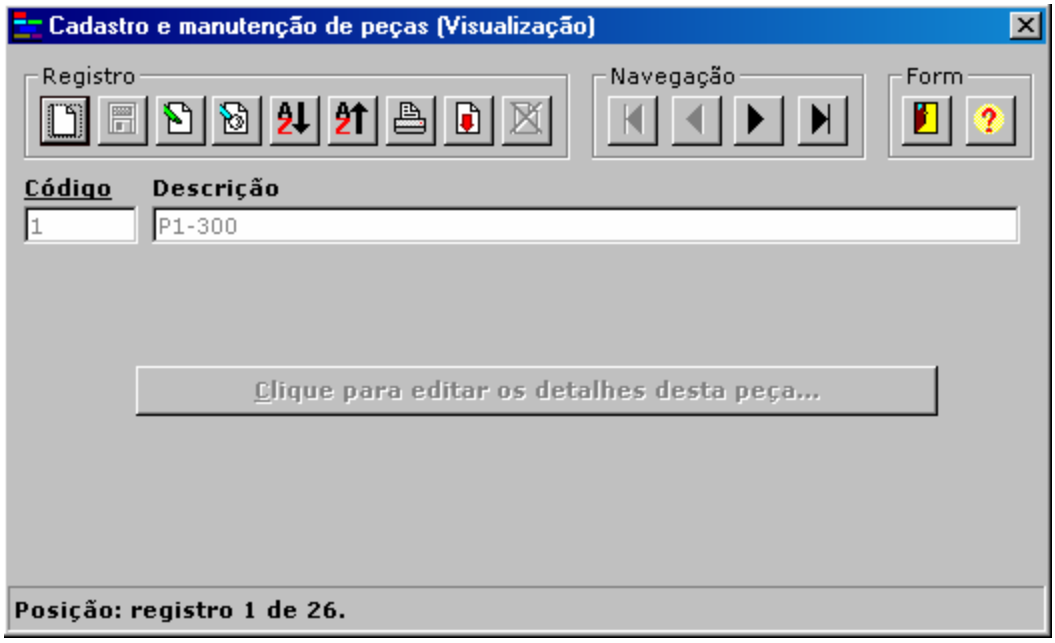


Figura 11: Cadastro e manutenção de peças no SIPMA.

O detalhamento do cadastro de peças deve ser feito com cuidado pois envolve informações fundamentais para a execução do modelo. Ativando-se o botão "*Clique para editar os detalhes desta peça...*" surge o formulário que dá acesso aos dados sobre **demanda**, **lotes**, **roteiro** e **representação na carta de Gantt** que cada uma possui (Figuras de 12 a 16).

A quantidade de peças a serem produzidas é especificada na guia *Demandas*. Convencionou-se defini-la mês a mês a partir de um valor base único sobre o qual todos os demais são calculados automaticamente, seguindo-se um perfil pré-estabelecido que consiste em variações percentuais deste valor (Figura 12). A edição do perfil ocorre pelo botão "*Editar perfil...*". No formulário apropriado (Figura 13) o usuário deverá informar o percentual acima ou abaixo do valor base em cada mês. Retornando à tela anterior, a ativação do botão "*Calcular demandas*" preenche as caixas de texto com as quantidades adequadas. Caso seja necessário, é possível também digitá-las diretamente na caixa do mês desejado.

Figura 12: Determinação da demanda básica e seu perfil anual.

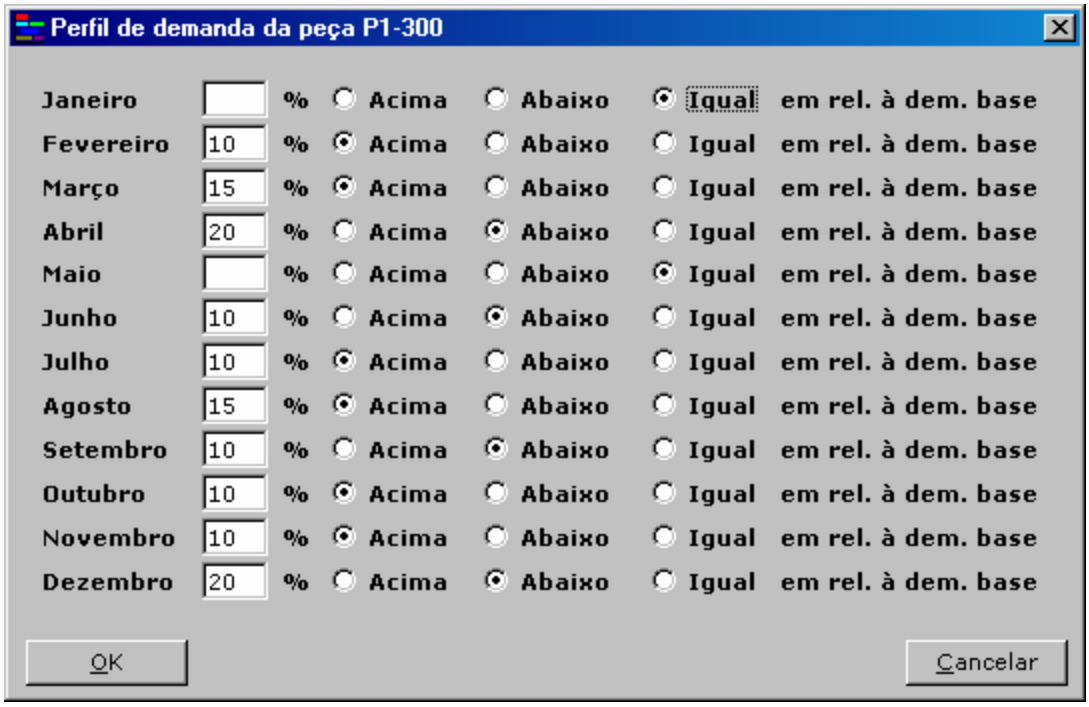


Figura 13: Perfil de demanda.

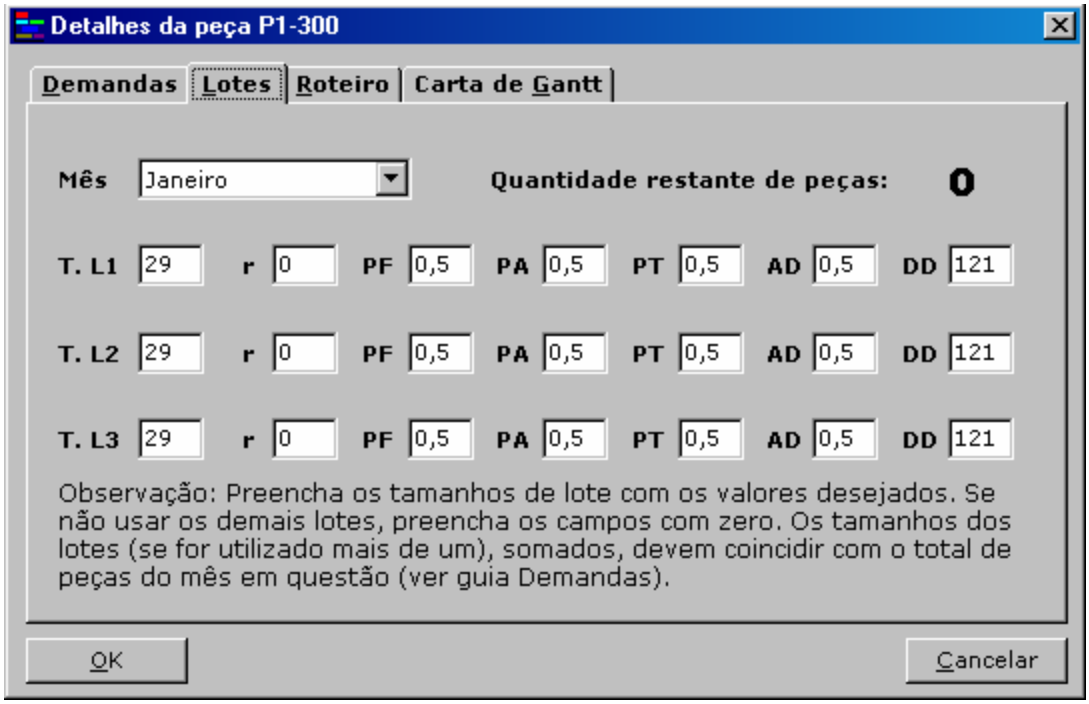


Figura 14: Informações sobre o lote em um dado mês (janeiro).

Na guia *Lotes* (Figura 14) é possível dividir a demanda de uma peça em até três conjuntos menores cujos tamanhos não podem ultrapassar a quantidade máxima definida em um dado mês. No exemplo acima, observa-se que a peça P1-300 tem uma demanda no mês de janeiro de 87 unidades que devem ser seqüenciadas em três lotes de 29 peças cada.

Além do tamanho, cada lote deve ter definidas as seguintes variáveis:

- Tempo de entrada no sistema (r_j): o instante de tempo em que o lote iniciará seu processamento. Em casos estáticos, $r_j = 0$, para qualquer j;
- PF: peso por fluxo; é o fator de ponderação w_j utilizado nas funções-objetivo definidas pelas equações [6] e [7] (capítulo 3, páginas 37 e 38);
- PA: peso por atraso; é o fator de ponderação w_{Tj} utilizado nas funções-objetivo definidas pelas equações [8] e [10] (capítulo 3, página 38);
- PT: peso por adiantamento; é o fator de ponderação w_{Ej} utilizado na função-objetivo definida pela equação [10] (capítulo 3, página 38);

- AD: peso por número de atividades adiadas; é o fator de ponderação w_{Nj} utilizado na função-objetivo definida pela equação [9] (capítulo 3, página 38);
- DD (*due date*): data-limite de entrega; prazo máximo que o lote tem para ser despachado dentro de um horizonte de tempo. Dependendo da heurística e da função-objetivo esta variável pode influenciar positiva ou negativamente no resultado final do seqüenciamento.

Incorporadas estas informações, deve-se seguir para a próxima guia – *Roteiro* – em que será construída a rota de fabricação da peça (Figura 15):

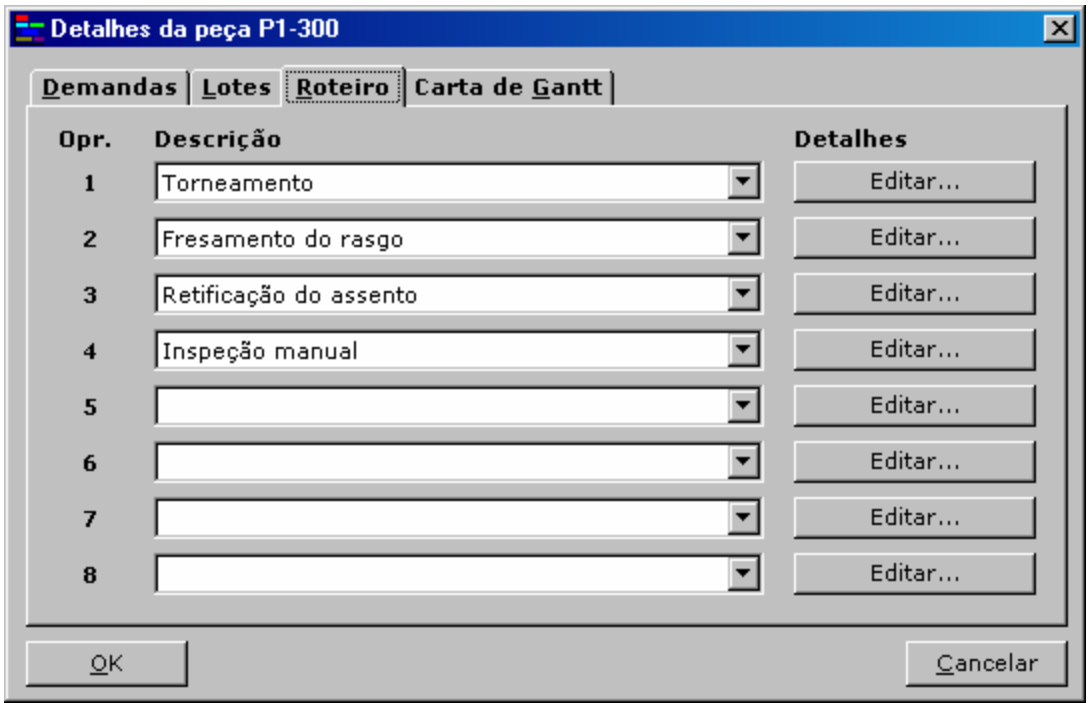


Figura 15: Roteiro de fabricação no SIPMA.

Um roteiro pode ter até 8 processos pelos quais a peça passará. No exemplo da Figura 15 a peça P1-300 é, em primeiro lugar, torneada, depois executa-se sobre ela uma operação de fresamento de rasgo, em seguida uma retificação, sendo inspecionada manualmente ao final de sua produção. Os processos têm as máquinas que os executam atribuídas por meio do botão "*Editar...*", presente à frente de cada um (Figura 16).

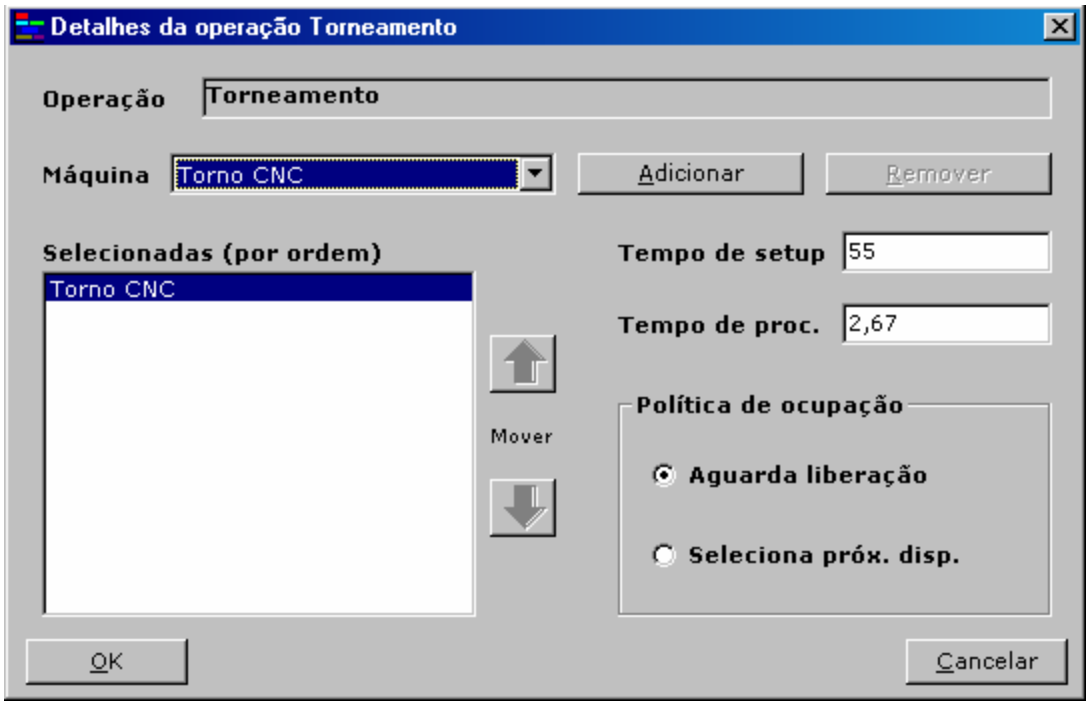


Figura 16: Detalhamento do roteiro de fabricação.

O exemplo ilustra o detalhamento da operação de torneamento da peça P1-300. Deve-se selecionar todas as máquinas que podem desempenhar esta função na caixa suspensa "*Máquina*", adicionando-as à lista. Nesta, a ordem é importante: a primeira é a padrão, a escolhida para desempenhar a operação atual assim que a peça for liberada do processo anterior. Por ocasião do seqüenciamento, caso a máquina principal esteja em uso é possível tomar duas decisões: esperar que ela seja liberada ou redirecionar a peça para qualquer uma das *máquinas alternativas* escolhidas, que são as opções seguintes da lista, se existirem. Para isso, deve-se marcar a opção desejada na caixa "*Política de ocupação*". Se for definida apenas uma máquina, a peça iniciará seu processamento apenas quando a mesma se encontrar liberada; se mais de uma for definida e todas estiverem ocupadas, aguarda-se até que haja uma disponível. As setas ao lado da caixa de lista podem ser utilizadas para trocar a ordem de prioridade das máquinas relacionadas ao processo.

Em seguida deve-se determinar o tempo de *setup* (preparação) e o tempo de processamento da peça em cada máquina. Estes dados servirão para calcular, na criação do cenário (item **4.1.1.2 Criação do cenário de resolução**) o tempo **total** de processamento por meio da expressão:

$$\left(SM/TLote\right) + TProc \qquad [32]$$

Nesta equação tem-se:

- SM = tempo de *setup*;
- TLote = tamanho do lote;
- TProc = tempo de processamento de uma unidade da peça

Por exemplo, para um lote de 29 unidades da peça P1-300 a serem processadas num torno cujo tempo de *setup* é de 55 minutos e o tempo de processamento é 2,67 minutos, determina-se que o tempo total de processamento da peça é:

$$\left(55/29\right) + 2{,}67 = 4{,}56\,\text{min}$$

O tempo total de processamento do lote será, portanto, de 132 (29 x 4,56) minutos, aproximadamente, ou 2,2 horas.

A expressão $\left(SM/TLote\right)$ representa um *rateio* do tempo de *setup* pelo tamanho do lote. Isto significa que dependendo da quantidade de peças a serem processadas o tempo de *setup* variará de acordo com esta proporção.

A conclusão deste procedimento encerra o fornecimento das informações essenciais ao funcionamento do SIPMA.

4.1.1.2 Criação do cenário de resolução

O passo seguinte consiste na criação do cenário em que o problema será resolvido. A Figura 17 exibe o formulário (chamado pelo menu *Cenário*, opção *Criar...*) em que esta função ocorre:

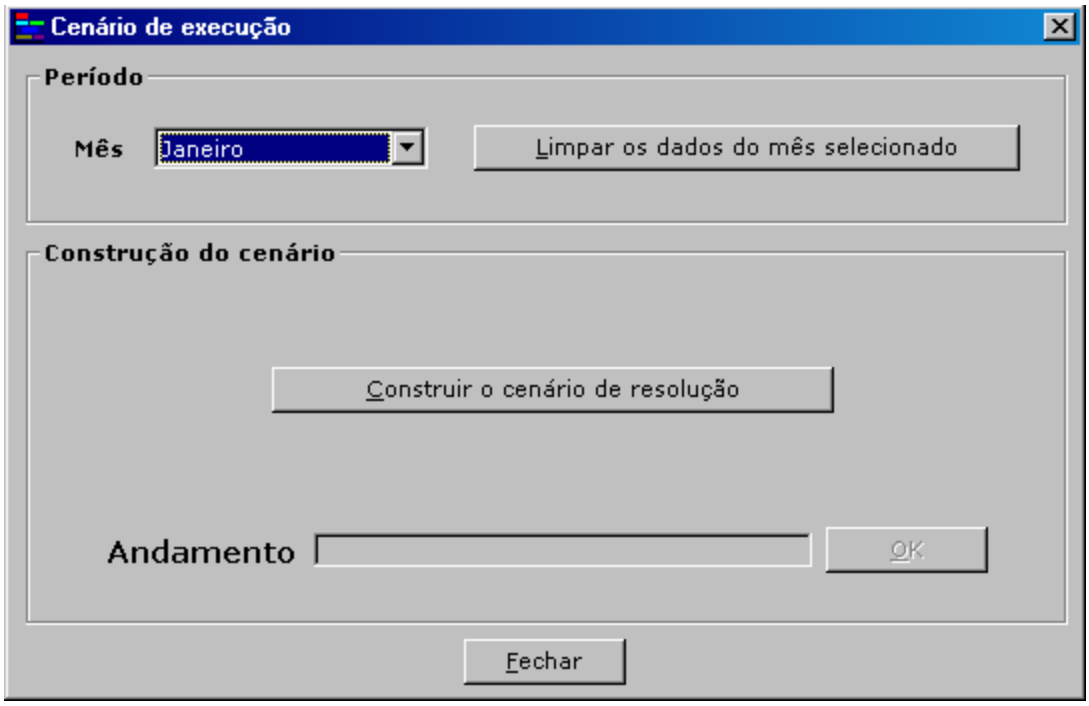


Figura 17: Construção do cenário de execução.

Um *cenário* é um conjunto de condições iniciais obtidas pelo processamento dos dados armazenados sobre máquinas, operações e peças, num dado mês. Serão utilizadas pelas heurísticas para determinar, pelo processo iterativo descrito à página 35, a melhor seqüência de produção da situação em estudo.

Escolhido o mês desejado (pela caixa suspensa apropriada), a ativação do botão "*Construir o cenário de resolução*" gera e armazena as informações adequadamente. O tópico 4.1.4 (**Fluxo de dados no SIPMA**) apresentará maiores detalhes sobre a natureza dos dados que compõem o cenário e como este fornece os subsídios necessários para a construção das cartas de Gantt, das medidas de desempenho e dos valores das funções-objetivo.

4.1.1.3 Execução do modelo

Construído o ambiente da simulação, finaliza-se a parte de resolução do problema ativando-se o menu "*Modelo*", opção "*Executar cenário...*". Surge o formulário em que as respostas serão calculadas após a escolha da heurística (Figura 18):

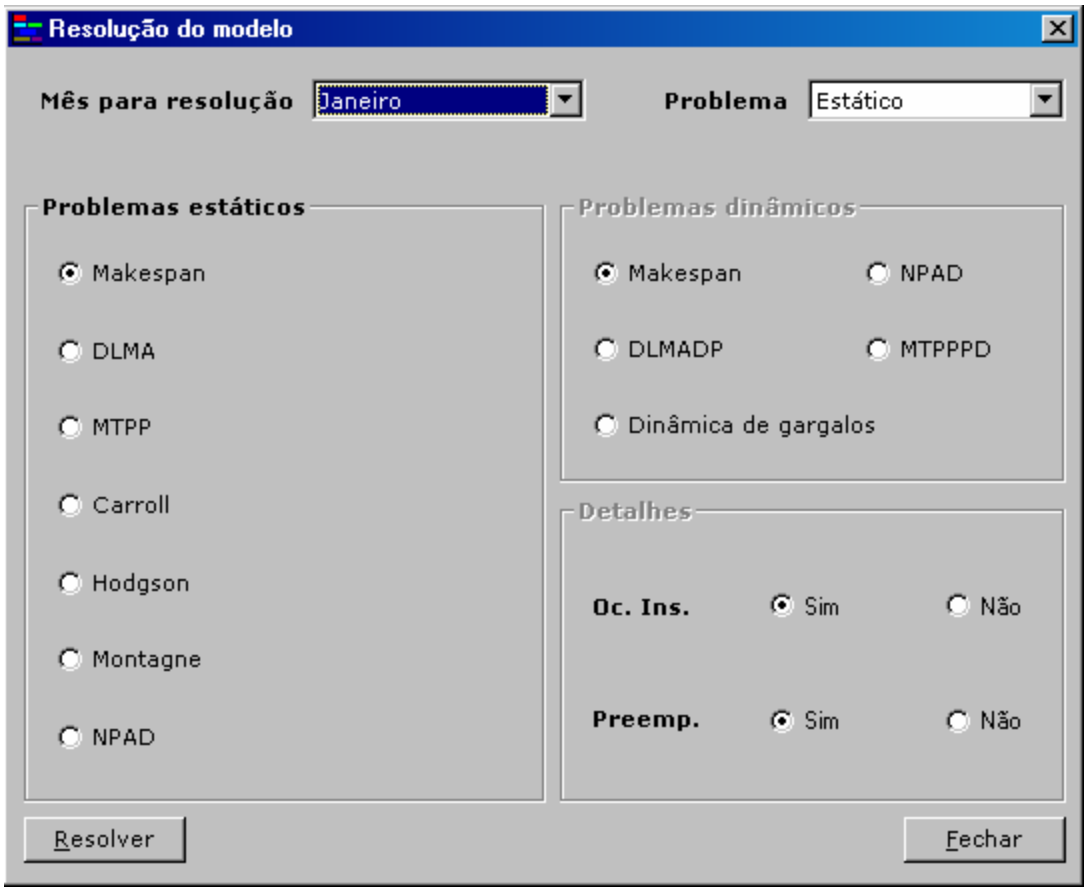


Figura 18: Escolha do modelo de resolução.

Após a conclusão do algoritmo selecionado estará disponível ao usuário o resultado, por meio do menu "*Resultados*", opção "*Modo texto...*" (exibição dos nomes das peças, lotes e seus respectivos tempos de início e fim de processamento, numa tabela) ou "*Carta de Gantt...*" (visualização gráfica do seqüenciamento via blocos coloridos ordenados no tempo, por máquina, que indicam os momentos em que a peça inicia e finaliza um processamento).

4.1.2 Especificações de banco de dados do *software* SIPMA

Como interface entre o usuário e um conjunto de dados (as rotinas heurísticas de ordenação, os dados inerentes ao problema e suas respostas) é função do SIPMA administrar sua edição (inclusão, alteração e exclusão), recuperação, exibição e uso, sempre de modo transparente ao usuário. O banco de dados escolhido (Microsoft® Access 97 , pertencente ao pacote Microsoft® Office 97) é conveniente dada sua facilidade de uso e o poder de manipulação

de conjuntos extensos de dados, numa plataforma computacional bem estabelecida (Microsoft® Windows 98™)

Existem no total 9 tabelas com as quais o SIPMA trabalha escrevendo ou recuperando dados. Suas finalidades encontram-se discriminadas a partir da página seguinte.

Tabelas de armazenamento de dados primários (sobre máquinas, operações e peças):

Máquinas: responsável pelo armazenamento de informações referentes às máquinas.

Estrutura	
Nome do campo	**Descrição**
Maq_COD	Código da máquina
Maq_DES	Descrição da máquina
Maq_CPU	Custo de operação por unidade de tempo

Operações: responsável pelo armazenamento de informações referentes às operações.

Estrutura	
Nome do campo	**Descrição**
Opr_COD	Código da operação
Opr_DES	Descrição da operação
Opr_MAQ	Máquinas em que a operação pode ser desempenhada

Peças: responsável pelo armazenamento de informações referentes às peças.

Estrutura	
Nome do campo	**Descrição**
Pcs_COD	Código da peça
Pcs_DES	Descrição da peça

Pcs_PDM	Perfil de demanda da peça
Pcs_DEM	Demanda mensal da peça no ano
Pcs_LOT	Tamanho de lotes
Pcs_ROT	Roteiro de processamento
Pcs_DRO	Detalhamento do roteiro de processamento
Pcs_COR	Cor da peça em sua representação da carta de Gantt

Tabelas de armazenamento de dados para a solução do modelo:

Cenário: responsável pelo armazenamento de informações referentes ao cenário base para a execução do modelo.

Estrutura	
Nome do campo	**Descrição**
Cen_MES	Mês em que o lote da peça será seqüenciado
Cen_PCS	Código da peça
Cen_LOT	Lote da peça (1, 2 ou 3)
Cen_TLO	Tamanho do lote
Cen_INP	Início de processamento no sistema (o tempo em que o lote dá entrada no sistema)
Cen_PPF	Peso por fluxo do lote
Cen_PPA	Peso por adiamento do lote
Cen_PPT	Peso por adiantamento do lote
Cen_PAD	Peso por atividade adiada do lote
Cen_DDT	*Due date* (data limite) de entrega do lote
Cen_COR	Cor da peça em sua representação da carta de Gantt
Cen_OPT	Roteiro desta peça com seus respectivos tempos finais de processamento

Resposta: responsável pelo armazenamento da resposta obtida pela resolução do modelo, utilizando a heurística escolhida.

Estrutura	
Nome do campo	**Descrição**
Res_MES	Mês de seqüenciamento
Res_HEU	Heurística utilizada para a resolução do modelo
Res_PCS	Código da peça (apenas para visualização da ordem)
Res_LOT	Lote (apenas para visualização da ordem)
Res_PDS	Descrição final da peça (código da peça em dois dígitos + lote em dois dígitos)
Res_ORP	Ordinal para a seqüência das operações
Res_MAQ	Código da máquina em que a operação será desempenhada
Res_TPR	Tempo de processamento
Res_POL	Política de ocupação
Res_MAL	Máquinas alternativas e respectivos tempos de processamento para aquela operação
Res_INP	Início da operação no sistema
Res_DDT	*Due date* da entrega do lote
Res_COR	Cor da peça em sua representação da carta de Gantt

Gantt: responsável pelo armazenamento de informações para a construção da carta de Gantt.

Estrutura	
Nome do campo	**Descrição**
Gnt_MES	Mês de seqüenciamento
Gnt_HEU	Heurística utilizada para a resolução do modelo
Gnt_PCS	Código da peça
Gnt_LOT	Lote

Gnt_PDS	Descrição final da peça (código da peça em dois dígitos + lote em dois dígitos)
Gnt_ORP	Ordinal para a seqüência das operações
Gnt_MAQ	Código da máquina em que a operação será desempenhada
Gnt_TIN	Tempo de início de processamento
Gnt_TFM	Tempo de final de processamento
Gnt_INP	Início da operação no sistema
Gnt_DDT	*Due date* da entrega do lote
Gnt_COR	Cor da representação da peça na carta de Gantt

Tabelas de armazenamento de dados pós-resolução do modelo:

Custos: responsável pelo armazenamento de informações referentes aos tempos de utilização das máquinas e seus custos totais, em cada execução de uma heurística.

Estrutura	
Nome do campo	**Descrição**
Cus_COD	Código da máquina
Cus_MES	Mês em que os custos foram computados
Cus_TPU	Tempo de uso total da máquina naquele mês
Cus_CUS	Custo total de uso da máquina naquele mês
Cus_HEU	Heurística que levou a esses custos

Medidas: responsável pelo armazenamento de informações referentes às medidas de desempenho definidas às páginas 35 e 36.

Estrutura	
Nome do campo	**Descrição**
Med_MES	Mês de seqüenciamento
Med_HEU	Heurística utilizada para a resolução do modelo

Med_PCS	Código da peça
Med_LOT	Lote
Med_PDS	Descrição final da peça (código da peça em dois dígitos + lote em dois dígitos)
Med_TCO	Tempo de completamento (*completion time*) (C_j)
Med_TFL	Tempo de fluxo (F_j)
Med_ATR	Atraso (L_j)
Med_ADI	Adiamento (T_j)
Med_ADN	Adiantamento (E_j)
Med_PPF	Peso por fluxo do lote (w_j)
Med_PPA	Peso por adiamento do lote (w_{Tj})
Med_PPT	Peso por adiantamento do lote (w_{Ej})
Med_PAD	Peso por atividade adiada do lote (w_{Nj})

Funções: responsável pelo armazenamento de informações referentes às funções-objetivo definidas às páginas 37-39.

Estrutura	
Nome do campo	**Descrição**
Fun_MES	Mês de seqüenciamento
Fun_HEU	Heurística utilizada para a resolução do modelo
Fun_MKS	Makespan (C_{max})
Fun_TFP	Tempo de fluxo ponderado (F_{wt})
Fun_ATP	Atraso ponderado (L_{wt})
Fun_ADP	Adiamento ponderado (T_{wt})
Fun_NPA	Número ponderado de atividades adiadas (N_{wt})
Fun_AAP	Adiantamento ponderado + adiamento ponderado (ET_{wt})
Fun_TMF	Tempo máximo de fluxo (F_{max})
Fun_ TMA	Tempo máximo de atraso (L_{max})
Fun_TMD	Tempo máximo de adiamento (T_{max})

4.1.3 Metodologia de armazenamento e recuperação de dados

O SIPMA emprega em sua estrutura interna de armazenamento e recuperação de dados a codificação e decodificação de informações por meio *strings*. Uma *string* é uma cadeia de caracteres cujas posições podem ser lidas separadamente. É possível empregar funções adequadas no ambiente de desenvolvimento (linguagem de programação) para ler partes da *string* (desde apenas um caractere até todos que a formam), substituir um trecho de uma *string* por um trecho de outra, determinar se determinado padrão de caracteres se encontra dentro de uma *string*, entre outras. Pode-se ilustrar o processo pelo campo **Opr_MAQ** (tabela *Operações*) que armazena uma *string* no seguinte formato:

QQCM1PL1CM2PL2CM3PL3...

Nesta, define-se:

QQ = quantidade de máquinas que podem executar aquela operação (2 dígitos)
CM1 = Máquina 1 (código); PL1 = Uso interno do SIPMA (ambos com 3 dígitos)
CM2 = Máquina 2 (código); PL2 = Uso interno do SIPMA (ambos com 3 dígitos)
CM3 = Máquina 3 (código); PL3 = Uso interno do SIPMA (ambos com 3 dígitos)...

O total de máquinas nos dois primeiros caracteres da *string* determinará o tamanho final da mesma, que é *2 + 6 * (Quantidade de máquinas)*. Assim, um exemplo possível de conteúdo do campo **Opr_MAQ** é:

04001003003005002004007002

Em que, na seqüência:

04 = quantidade de máquinas que podem executar aquela operação (4)
001 = Código da máquina: 1; 003 = Uso interno do SIPMA;
003 = Código da máquina: 3; 005 = Uso interno do SIPMA;

002 = Código da máquina: 2; 004 = Uso interno do SIPMA;
007 = Código da máquina: 7; 002 = Uso interno do SIPMA;

Tamanho da *string*: 26 posições (caracteres) = 2 + 6*4.

Neste caso em particular não é necessário levar em conta a ordem das máquinas e seu número de controle interno. Deste modo, é indiferente que se armazene no campo **Opr_MAQ** um conteúdo cujo formato seja *040030050070020010030020004*, *040070020020040010030030005* ou *040020040010030030050070002*, desde que os dois primeiros caracteres indiquem sempre a quantidade de máquinas que podem desempenhar a operação em edição.

A dinâmica de armazenamento e recuperação segue o princípio:

Entrada do usuário ⟶ Codificação da *string* ⟶ Armazenamento

e

Banco de dados ⟶ Decodificação da *string* ⟶ Exibição / edição / uso interno

Por *uso interno* deve-se entender a decodificação da *string* para determinar, por exemplo, tempos de processamento, máquina a ser utilizada ou próxima operação a ser desempenhada.

O processo geral de codificação / decodificação consiste em montar cadeias de caracteres com as entradas do usuário, seguindo padrões previamente estabelecidos. Conhecendo a estrutura da cadeia de caracteres (ver Anexo I) é possível, portanto, "desmontá-la" e recuperar cada informação individual que pode ser tanto exibida para o usuário como utilizada na resolução de um dado modelo. O SIPMA incorpora rotinas que executam as duas funções, sendo que cada uma é diferente das demais, pelo fato de cada cadeia possuir uma estrutura diferente. Deste modo foi possível criar um sistema enxuto, facilmente modificável para implementações futuras.

4.1.4 Fluxo de dados no SIPMA

A figura a seguir (Figura 19) mostra como são tratadas as informações pelo SIPMA:

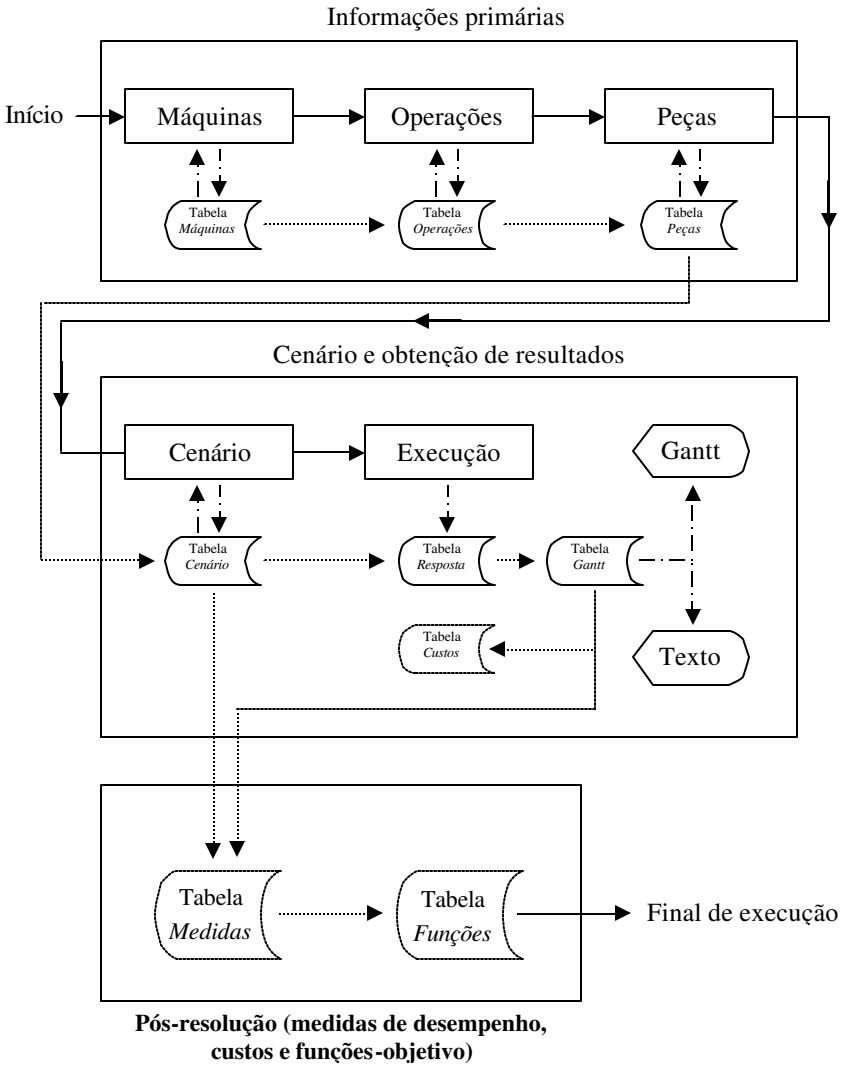


Figura 19: Fluxo de dados no SIPMA.

Neste diagrama as setas de linha cheia indicam o caminho que se deve percorrer do cadastramento de dados, escolha da heurística, execução do modelo e obtenção de resultados até o final da execução do sistema. As setas de traço-e-ponto indicam a interação entre um formulário (que é o elo de comunicação entre o usuário e o banco de dados) e sua respectiva tabela (por meio dele pode-se alterar os dados armazenados ou visualizá-los). Quando não há

uma ligação direta entre ambos, significa que a mesma é alterada indiretamente pelo SIPMA como resultado de um ou mais processamentos de dados: isto é representado pelas setas de linha pontilhada. As tabelas abaixo (Tabelas 2 e 3) resumem este fluxo de informações:

Tabela 2: Alteração de dados pelos formulários do SIPMA.

O formulário	Modifica os dados da(s) tabela(s)...[1]	Operações possíveis
Máquinas	Máquinas (usuário)	Leitura / escrita
Operações	Operações (usuário)	Leitura / escrita
Peças	Peças (usuário)	Leitura / escrita
Cenário	Cenário (usuário)	Leitura / escrita
Execução	Resposta (SIPMA)	Leitura / escrita
	Gantt (SIPMA)	Leitura / escrita
	Custos (SIPMA)	Leitura / escrita
	Medidas (SIPMA)	Leitura / escrita
	Funções (SIPMA)	Leitura / escrita

Tabela 3: Dependência entre as tabelas no SIPMA.

A tabela	Fornece dados para a tabela...
Máquinas	Operações
Operações	Peças
Peças	Cenário
Cenário	Resposta, Medidas
Resposta	Gantt
Gantt	Custos, Medidas
Custos	(nenhuma)
Medidas	Funções
Funções	(nenhuma)

[1] Por "*modificar dados*" em uma tabela, entenda-se *inserir*, *alterar* ou *excluir* dados.

Na Tabela 2, as modificações provocadas pelo formulário *Execução* são resultado da ação de calcular as soluções do problema, por meio do botão "*Resolver*" (ver Figura 18): quando as mesmas são obtidas, o SIPMA as armazena na tabela *Resposta* que depois terá suas informações utilizadas pelas tabelas *Gantt*, *Custos*, *Medidas* e *Funções*, para a definição dos dados de análise e visualização gráfica do seqüenciamento. Em outras palavras, o *usuário* modifica as tabelas *Máquinas*, *Operações*, *Peças* e *Cenário*, ao passo que o SIPMA, com tais entradas, modifica as tabelas *Resposta*, *Gantt*, *Custos*, *Medidas* e *Funções*.

Com relação à dependência entre tabelas (Tabela 3), ocorre o seguinte:

- A tabela *Operações* armazenará informações sobre as máquinas que podem executar cada uma das operações cadastradas, por meio de uma *string* apropriada (ver páginas 75 e 76 e Anexo I);
- A tabela *Peças*, por sua vez, armazenará dados sobre o roteiro de fabricação das peças com base na *string* definida no item anterior bem como armazenará os dados sobre demanda e lotes, igualmente por meio de uma *string*;
- A tabela *Cenário* armazenará cada um dos tamanhos de lote, pesos, tempos de entrada no sistema e datas-limite de cada peça. Essas informações são obtidas pela decodificação das *strings* lidas da tabela *Peças*. Também guardará uma *string* de roteiro final, semelhante à obtida no item anterior, incluindo os tempos totais de processamento de cada peça;
- A tabela *Resposta* será preenchida após a leitura e processamento da tabela *Cenário* pela regra heurística escolhida para a resolução do problema, que ordenará as atividades corretamente, fornecendo, simultaneamente, recursos para a construção da carta de Gantt, dos custos de utilização das máquinas, das medidas de desempenho e dos valores das funções-objetivo, completando o ciclo de funcionamento do SIPMA.

4.2 Um teste prático – produção de peças numa célula de manufatura

Serão apresentados os dados referentes a uma linha de produção de peças, que constituirá o objeto de estudo deste trabalho. Dispõe-se de uma célula de manufatura montada no Laboratório

de Usinagem da Faculdade de Engenharia Mecânica – UNICAMP e o objetivo é programar a fabricação de um conjunto de peças desenvolvidas expressamente para o modelo.

O laboratório conta com três máquinas principais: um Centro de Torneamento, um Centro de Usinagem e um Centro de Retificação em que ocorre o fluxo de produção (Figura 20). As máquinas são todas controladas por comando numérico (CNC). Especificam-se os dados:

- Dias de funcionamento por mês: 20 (4 semanas);
- Dias por semana: 5;
- Quantidade de horas de funcionamento por dia: 9.

Tal distribuição de tempo implica em 45 horas por semana. Esta informação é útil na interpretação da carta de Gantt resultante da execução do modelo, que exibe gráficos agrupados de 45 em 45 horas, para cada semana, em cada mês.

A unidade de tempo no SIPMA é o minuto para *setup* e tempos de processamento e a hora para datas-limite e tempos de entrada no sistema (para problemas dinâmicos); assim, todos os campos devem ser preenchidos adequadamente, visando correto funcionamento do sistema.

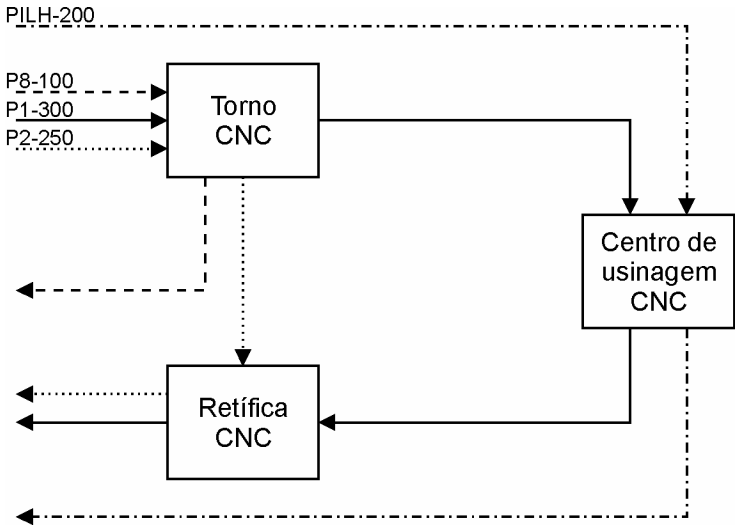


Figura 20: Exemplo de fluxo de peças pela célula do laboratório de usinagem.

A seguir, são apresentados os dados criados para testar o *software* e as heurísticas escolhidas. São 26 peças agrupadas nas famílias E1, E2, E3, B1, B2, S1, S2. Ao lado do nome da peça está seu respectivo código numérico associado quando de seu cadastramento no SIPMA. O Anexo II apresenta em maiores detalhes a conformação e os dados parametrizados de cada peça.

Família E1:

- Peça P1-300 (código da peça no SIPMA: 01)

(Tempos de Setup: Torno = 55 min; Ret = 30 min e CU = 15 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	2,67
20	Fresamento rasgo	CU-CNC	5,00
30	Retificação de assento	RET-CNC	0,27
40	Inspeção manual	BANCADA	0,50

- Peça P10-280 (código da peça no SIPMA: 02)

(Tempos de Setup: Torno = 55 min; Ret = 30 min e CU = 15 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	2,40
20	Fresamento rasgo	CU-CNC	4,80
30	Retificação de assento	RET-CNC	0,25
40	Inspeção manual	BANCADA	0,50

- Peça P20-250 (código da peça no SIPMA: 03)

(Tempos de Setup: Torno = 55 min; Ret = 30 min e CU = 15 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	2,30
20	Fresamento rasgo	CU-CNC	4,50
30	Retificação de assento	RET-CNC	0,24
40	Inspeção manual	BANCADA	0,50

- Peça P30-200 (código da peça no SIPMA: 04)

(Tempos de Setup: Torno = 55 min; Ret = 30 min e CU = 15 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	2,20
20	Fresamento rasgo	CU-CNC	4,00
30	Retificação de assento	RET-CNC	0,24
40	Inspeção manual	BANCADA	0,5

- Peça P40-150 (código da peça no SIPMA: 05)

(Tempos de Setup: Torno = 55 min; Ret = 30 min e CU = 15 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	2,12
20	Fresamento rasgo	CU-CNC	3,80
30	Retificação de assento	RET-CNC	0,21
40	Inspeção manual	BANCADA	0,50

Família E2 (2 fixações):

- Peça P2-250 (código da peça no SIPMA: 06)

(Tempos de Setup: Torno = 55 min; Ret = 30 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,87
20	Retificar assento	RET-CNC	0,13
30	Inspeção manual	BANCADA	0,45

- Peça P3-200 (código da peça no SIPMA: 07)

(Tempos de Setup: Torno = 55 min; Ret = 30 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,61
20	Retificar assento	RET-CNC	0,12
30	Inspeção manual	BANCADA	0,45

- Peça P4-150 (código da peça no SIPMA: 08)

(Tempos de Setup: Torno = 55 min; Ret = 30 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,35
20	Retificar assento	RET-CNC	0,11
30	Inspeção manual	BANCADA	0,45

- Peça P5-100 (código da peça no SIPMA: 09)

(Tempos de Setup: Torno = 55 min; Ret = 30 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,20
20	Retificar assento	RET-CNC	0,10
30	Inspeção manual	BANCADA	0,45

Família E3 (2 fixações):

- Peça P6-150 (código da peça no SIPMA: 10)

(Tempos de Setup: Torno = 55 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,60
20	Inspeção manual	BANCADA	0,40

- Peça P7-120 (código da peça no SIPMA: 11)

(Tempos de Setup: Torno = 55 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,45
20	Inspeção manual	BANCADA	0,40

- Peça P8-100 (código da peça no SIPMA: 12)

(Tempos de Setup: Torno = 55 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,20
20	Inspeção manual	BANCADA	0,40

Família B1 (2 fixações c/ rasgo):

- Peça P11-100 (código da peça no SIPMA: 13)

(Tempos de Setup: Torno = 45 min; CU = 30 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	2,20
20	Fresamento do rasgo	CU-CNC	5,30
30	Inspeção manual	BANCADA	0,35

- Peça P12-90 (código da peça no SIPMA: 14)

(Tempos de Setup: Torno = 45 min; CU = 30 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	2,12
20	Fresamento do rasgo	CU-CNC	5,00
30	Inspeção manual	BANCADA	0,35

- Peça P13-80 (código da peça no SIPMA: 15)

(Tempos de Setup: Torno = 45 min; CU = 30 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,94
20	Fresamento do rasgo	CU-CNC	4,8
30	Inspeção manual	BANCADA	0,35

- Peça P14-70 (código da peça no SIPMA: 16)

(Tempos de Setup: Torno = 45 min; CU = 30 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	1,82
20	Fresamento do rasgo	CU-CNC	4,5
30	Inspeção manual	BANCADA	0,35

Família B2 (2 fixações):

- Peça P21-70 (código da peça no SIPMA: 17)

(Tempos de Setup: Torno = 55 min; Ret = 25 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	7,20
20	Retificar assento	RET-CNC	0,25
30	Inspeção manual	BANCADA	0,50

- Peça P22-60 (código da peça no SIPMA: 18)

(Tempos de Setup: Torno = 55 min; Ret = 25 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	6,30
20	Retificar assento	RET-CNC	0,24
30	Inspeção manual	BANCADA	0,50

- Peça P23-50 (código da peça no SIPMA: 19)

(Tempos de Setup: Torno = 55 min; Ret = 25 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	5,80
20	Retificar assento	RET-CNC	0,24
30	Inspeção manual	BANCADA	0,50

- Peça P24-40 (código da peça no SIPMA: 20)

(Tempos de Setup: Torno = 55 min; Ret = 25 min)

Operação	Descrição	Máquina	Tempo (min)
10	Torneamento	TORNO-CNC	5,30
20	Retificar assento	RET-CNC	0,23
30	Inspeção manual	BANCADA	0,50

Família S1 (ilha):

- Peça PILH-300 (código da peça no SIPMA: 21)

(Tempos de Setup: CU = 40 min)

Operação	Descrição	Máquina	Tempo (min)
10	Fresamento da Ilha	CU-CNC	8,00
20	Inspeção manual	BANCADA	0,30

- Peça PILH-250 (código da peça no SIPMA: 22)

(Tempos de Setup: CU = 40 min)

Operação	Descrição	Máquina	Tempo (min)
10	Fresamento da Ilha	CU-CNC	7,20
20	Inspeção manual	BANCADA	0,30

- Peça PILH-200 (código da peça no SIPMA: 23)

(Tempos de Setup: CU = 40 min)

Operação	Descrição	Máquina	Tempo (min)
10	Fresamento da Ilha	CU-CNC	6,30
20	Inspeção manual	BANCADA	0,30

Família S2 (suporte plano):

- Peça P41-100 (código da peça no SIPMA: 24)

(Tempos de Setup: CU = 60 min)

Operação	Descrição	Máquina	Tempo (min)
10	Fresamento dos rasgos do suporte	CU-CNC	18,00
20	Inspeção manual	BANCADA	0,30

- Peça P42-90 (código da peça no SIPMA: 25)

(Tempos de Setup: CU = 60 min)

Operação	Descrição	Máquina	Tempo (min)
10	Fresamento dos rasgos do suporte	CU-CNC	15,00
20	Inspeção manual	BANCADA	0,30

- Peça P43-80 (código da peça no SIPMA: 26)

(Tempos de Setup: CU = 60 min)

Operação	Descrição	Máquina	Tempo (min)
10	Fresamento dos rasgos do suporte	CU-CNC	13,00
20	Inspeção manual	BANCADA	0,30

Na Tabela 4 a seguir são apresentados os dados da demanda básica.

Tabela 4: Variações do valor base indicando as demandas por mês.

Peça	M (demanda)
P1-300	**87**
P10-280	**87**
P20-250	**64**
P30-200	**79**
P40-150	**87**
P2-250	**110**
P3-200	**116**
P4-150	**143**
P5-100	**217**
P6-150	**145**
P7-120	**181**
P8-100	**181**
P11-100	**72**
P12-90	**101**
P13-80	**87**
P14-70	**72**
P21-70	**28**
P22-60	**14**
P23-50	**65**
P24-40	**58**

PILH-300	**37**
PILH-250	**37**
PILH-200	**43**
P41-100	**14**
P42-90	**22**
P43-80	**23**

Como já mencionado, os *setups* de todas as máquinas devem ser rateados pelos tamanhos dos lotes a serem processados.

A Tabela 5 a seguir exibe os perfis de demanda mês a mês, com relação às quantidades básicas estabelecidas na tabela anterior:

Tabela 5: Perfil de demanda mês a mês.

Mês	Valor da Demanda
1	**M (Valor da tabela)**
2	**10% acima de M**
3	**15% acima de M**
4	**20% abaixo de M**
5	**M (Valor da tabela)**
6	**10% abaixo de M**
7	**10% acima de M**
8	**15% acima de M**
9	**10% abaixo de M**
10	**10% acima de M**
11	**10% acima de M**
12	**20% abaixo de M**

Como se pode notar, o perfil de demanda apresenta períodos de maior ocupação da planta (entre 10% e 15% a mais) e períodos de menor ocupação (até 20% a menos). Serão escolhidos os meses de janeiro, março e dezembro (meses 1, 3 e 12 respectivamente) para os testes, uma vez

que os desvios de demanda dos outros meses são semelhantes. Estes meses podem, pois, ser referenciados para prever os testes para os demais períodos.

Em termos de heurísticas serão utilizadas as regras estáticas baseadas em *makespan*, em DLMA e em MTPP. A escolha se deve ao fato de que tais regras são as fontes para os demais casos estáticos e dinâmicos e, deste modo, a análise dos resultados gerados por estas regras pode levar a idéias sobre como as demais agirão.

Por fim, seguem os custos de utilização, por hora, das capacidades produtivas contidas na célula:

- Torno CNC: 20 unidades monetárias/h;
- Centro de usinagem CNC: 30 unidades monetárias /h;
- Centro de retificação CNC: 45 unidades monetárias /h;
- Bancada de inspeção: sem custo.

Capítulo 5

Resultados e discussões

De acordo com os dados estabelecidos no capítulo anterior foram gerados os dados que podem ser consultados a seguir e que serão a base para a operação do SIPMA. Para cada peça foram gerados até 3 lotes cuja soma das quantidades é igual à demanda total. Os meses-referência para os testes foram janeiro, março e dezembro por apresentarem demandas distintas; os demais meses ou tiveram demanda idêntica ou com pouca variação. Portanto, julgou-se conveniente executar os testes apenas para estes três períodos. É importante ressaltar que a divisão da demanda total em lotes menores *não é* efetuada pelo SIPMA: é uma decisão a ser tomada pelo planejador da produção, e é baseada, principalmente, na quantidade de tempo que os lotes empregam para serem processados, de modo a não ultrapassarem a capacidade diária da máquina utilizada.

A atribuição de pesos à produção das peças levou em conta sua quantidade no lote total, de acordo com o seguinte critério:

- De 1 a 100 peças – peso: 0,5;
- De 101 a 200 peças – peso: 1,0;
- De 201-300 peças – peso: 1,5;
- Acima de 301 peças – peso 2;

Para simplificar, tais valores serão utilizados para os quatro tipos de pesos definidos no capítulo 3; deste modo, se, por exemplo, a peça P1-300 deve ter 87 unidades produzidas em janeiro, então w_j = 0,5 (peso por fluxo), w_{Tj} = 0,5 (peso por adiamento), w_{Ej} = 0,5 (peso por adiantamento) e w_{Nj} = 0,5 (peso por número de atividades adiadas), o que será representado, genericamente, pela última coluna (P) nas Tabelas 6, 7 e 8 a seguir:

Tabela 6: Tamanhos dos lotes e peso para o mês de janeiro.

Peça	Lote 1	Lote 2	Lote 3	Total	P
P1-300	29	29	29	87	0,5
P10-280	29	29	29	87	0,5
P20-250	20	25	19	64	0,5
P30-200	25	31	23	79	0,5
P40-150	19	34	34	87	0,5
P2-250	43	25	42	110	1
P3-200	25	45	46	116	1
P4-150	43	44	56	143	1
P5-100	85	60	72	217	1,5
P6-150	46	53	46	145	1
P7-120	61	61	59	181	1
P8-100	59	61	61	181	1
P11-100	28	18	26	72	0,5
P12-90	39	30	32	101	1
P13-80	20	33	34	87	0,5
P14-70	29	25	18	72	0,5
P21-70	11	9	8	28	0,5
P22-60	5	5	4	14	0,5
P23-50	15	24	26	65	0,5
P24-40	23	23	12	58	0,5
P41-100	8	15	14	37	0,5
P42-90	10	17	10	37	0,5
P43-80	12	15	16	43	0,5
PILH-300	5	5	4	14	0,5
PILH-250	6	7	9	22	0,5
PILH-200	8	8	7	23	0,5

Tabela 7: Tamanhos dos lotes e peso para o mês de março.

Peça	Lote 1	Lote 2	Lote 3	Total	P
P1-300	34	34	34	102	1
P10-280	34	34	34	102	1
P20-250	25	25	25	75	0,5
P30-200	31	31	31	93	0,5
P40-150	34	34	34	102	1
P2-250	43	43	43	129	1
P3-200	45	45	46	136	1
P4-150	56	56	56	168	1
P5-100	85	85	85	255	1,5
P6-150	56	57	57	170	1
P7-120	71	71	71	213	1,5
P8-100	71	71	71	213	1,5
P11-100	28	28	29	85	0,5
P12-90	39	40	40	119	1
P13-80	34	34	34	102	1
P14-70	29	28	28	85	0,5
P21-70	11	11	11	33	0,5
P22-60	5	6	6	17	0,5
P23-50	25	26	26	77	0,5
P24-40	23	23	22	68	0,5
P41-100	14	15	14	43	0,5
P42-90	15	14	14	43	0,5
P43-80	17	17	17	51	0,5
PILH-300	6	6	5	17	0,5
PILH-250	8	9	9	26	0,5
PILH-200	9	9	9	27	0,5

Tabela 8: Tamanhos dos lotes e peso para o mês de dezembro.

Peça	Lote 1	Lote 2	Lote 3	Total	P
P1-300	23	23	23	69	0,5
P10-280	23	23	23	69	0,5
P20-250	17	17	17	51	0,5
P30-200	21	21	21	63	0,5
P40-150	23	23	23	69	0,5
P2-250	29	29	30	88	0,5
P3-200	30	31	31	92	0,5
P4-150	38	38	38	114	1
P5-100	57	58	58	173	1
P6-150	39	38	39	116	1
P7-120	48	48	49	145	1
P8-100	49	48	48	145	1
P11-100	19	20	19	58	0,5
P12-90	27	27	27	81	0,5
P13-80	23	23	23	69	0,5
P14-70	20	19	19	58	0,5
P21-70	8	7	7	22	0,5
P22-60	4	4	4	12	0,5
P23-50	17	18	17	52	0,5
P24-40	15	15	16	46	0,5
P41-100	10	10	9	29	0,5
P42-90	10	9	10	29	0,5
P43-80	11	12	12	35	0,5
PILH-300	4	4	4	12	0,5
PILH-250	6	6	6	18	0,5
PILH-200	6	6	6	18	0,5

As próximas três tabelas (Tabelas 9-12) exibem os tempos de processamento finais de cada peça, segundo a fórmula de *setup* rateado por lote (capítulo 4), após a geração do cenário:

Tabela 9: Tempos totais de processamento (em h) para os lotes do mês de janeiro.

Máquina	**Torno CNC**			**C. Us. CNC**			**Retífica**			**Inspeção**		
Peça / lote	**Lote 1**	**Lote 2**	**Lote 3**	**Lote 1**	**Lote 2**	**Lote 3**	**Lote 1**	**Lote 2**	**Lote 3**	**Lote 1**	**Lote 2**	**Lote 3**
P1-300	2,21	2,21	2,21	2,67	2,67	2,67	0,63	0,63	0,63	0,24	0,24	0,24
P10-280	2,08	2,08	2,08	2,57	2,57	2,57	0,62	0,62	0,62	0,24	0,24	0,24
P20-250	1,68	1,88	1,65	1,75	2,13	1,68	0,58	0,60	0,58	0,17	0,21	0,16
P30-200	1,83	2,05	1,76	1,92	2,32	1,78	0,60	0,62	0,59	0,21	0,26	0,19
P40-150	1,59	2,12	2,12	1,45	2,40	2,40	0,57	0,62	0,62	0,16	0,28	0,28
P2-250	2,26	1,70	2,23	0	0	0	0,59	0,55	0,59	0,32	0,19	0,32
P3-200	1,59	2,12	2,15	0	0	0	0,55	0,59	0,59	0,19	0,34	0,35
P4-150	1,88	1,91	2,18	0	0	0	0,58	0,58	0,60	0,32	0,33	0,42
P5-100	2,62	2,12	2,36	0	0	0	0,64	0,60	0,62	0,64	0,45	0,54
P6-150	2,14	2,33	2,14	0	0	0	0	0	0	0,31	0,35	0,31
P7-120	2,39	2,39	2,34	0	0	0	0	0	0	0,41	0,41	0,39
P8-100	2,10	2,14	2,14	0	0	0	0	0	0	0,39	0,41	0,41
P11-100	1,78	1,41	1,70	2,97	2,09	2,80	0	0	0	0,16	0,11	0,15
P12-90	2,13	1,81	1,88	3,75	3,00	3,17	0	0	0	0,23	0,18	0,19
P13-80	1,40	1,82	1,85	2,10	3,14	3,22	0	0	0	0,12	0,19	0,20
P14-70	1,63	1,51	1,30	2,68	2,38	1,85	0	0	0	0,17	0,15	0,11
P21-70	2,24	2,00	1,88	0	0	0	0,46	0,45	0,45	0,09	0,08	0,07
P22-60	1,44	1,44	1,34	0	0	0	0,44	0,44	0,43	0,04	0,04	0,03
P23-50	2,37	3,24	3,43	0	0	0	0,48	0,51	0,52	0,13	0,20	0,22
P24-40	2,95	2,95	1,98	0	0	0	0,50	0,50	0,46	0,19	0,19	0,10
P41-100	0	0	0	3,40	5,50	5,20	0	0	0	0,04	0,08	0,07
P42-90	0	0	0	3,50	5,25	3,50	0	0	0	0,05	0,09	0,05
P43-80	0	0	0	3,60	4,25	4,47	0	0	0	0,06	0,08	0,08
PILH-300	0	0	0	1,33	1,20	1,20	0	0	0	0,03	0,02	0,02
PILH-250	0	0	0	1,39	1,51	1,75	0	0	0	0,03	0,04	0,05
PILH-200	0	0	0	1,51	1,51	1,40	0	0	0	0,04	0,04	0,04

Tabela 10: Tempos totais de processamento (em h) para os lotes do mês de março.

Máquina	Torno CNC			C. Us. CNC			Retífica			Inspeção		
Peça / lote	Lote 1	Lote 2	Lote 3	Lote 1	Lote 2	Lote 3	Lote 1	Lote 2	Lote 3	Lote 1	Lote 2	Lote 3
P1-300	2,43	2,43	2,43	3,08	3,08	3,08	0,65	0,65	0,65	0,28	0,28	0,28
P10-280	2,28	2,28	2,28	2,97	2,97	2,97	0,64	0,64	0,64	0,28	0,28	0,28
P20-250	1,88	1,88	1,88	2,13	2,13	2,13	0,60	0,60	0,60	0,21	0,21	0,21
P30-200	2,05	2,05	2,05	2,32	2,32	2,32	0,62	0,62	0,62	0,26	0,26	0,26
P40-150	2,12	2,12	2,12	2,40	2,40	2,40	0,62	0,62	0,62	0,28	0,28	0,28
P2-250	2,26	2,26	2,26	0	0	0	0,59	0,59	0,59	0,32	0,32	0,32
P3-200	2,12	2,12	2,15	0	0	0	0,59	0,59	0,59	0,34	0,34	0,35
P4-150	2,18	2,18	2,18	0	0	0	0,60	0,60	0,60	0,42	0,42	0,42
P5-100	2,62	2,62	2,62	0	0	0	0,64	0,64	0,64	0,64	0,64	0,64
P6-150	2,41	2,44	2,44	0	0	0	0	0	0	0,37	0,38	0,38
P7-120	2,63	2,63	2,63	0	0	0	0	0	0	0,47	0,47	0,47
P8-100	2,34	2,34	2,34	0	0	0	0	0	0	0,47	0,47	0,47
P11-100	1,78	1,78	1,81	2,97	2,97	3,06	0	0	0	0,16	0,16	0,17
P12-90	2,13	2,16	2,16	3,75	3,83	3,83	0	0	0	0,23	0,23	0,23
P13-80	1,85	1,85	1,85	3,22	3,22	3,22	0	0	0	0,20	0,20	0,20
P14-70	1,63	1,60	1,60	2,68	2,60	2,60	0	0	0	0,17	0,16	0,16
P21-70	2,24	2,24	2,24	0	0	0	0,46	0,46	0,46	0,09	0,09	0,09
P22-60	1,44	1,55	1,55	0	0	0	0,44	0,44	0,44	0,04	0,05	0,05
P23-50	3,33	3,43	3,43	0	0	0	0,52	0,52	0,52	0,21	0,22	0,22
P24-40	2,95	2,95	2,86	0	0	0	0,50	0,50	0,50	0,19	0,19	0,18
P41-100	0	0	0	5,20	5,50	5,20	0	0	0	0,07	0,08	0,07
P42-90	0	0	0	4,75	4,50	4,50	0	0	0	0,08	0,07	0,07
P43-80	0	0	0	4,68	4,68	4,68	0	0	0	0,09	0,09	0,09
PILH-300	0	0	0	1,47	1,47	1,33	0	0	0	0,03	0,03	0,03
PILH-250	0	0	0	1,63	1,75	1,75	0	0	0	0,04	0,05	0,05
PILH-200	0	0	0	1,61	1,61	1,61	0	0	0	0,05	0,05	0,05

Tabela 11: Tempos totais de processamento (em h) para os lotes do mês de dezembro.

Máquina	Torno CNC			C. Us. CNC			Retífica			Inspeção		
Peça / lote	Lote 1	Lote 2	Lote 3	Lote 1	Lote 2	Lote 3	Lote 1	Lote 2	Lote 3	Lote 1	Lote 2	Lote 3
P1-300	1,94	1,94	1,94	2,17	2,17	2,17	0,60	0,60	0,60	0,19	0,19	0,19
P10-280	1,84	1,84	1,84	2,09	2,09	2,09	0,60	0,60	0,60	0,19	0,19	0,19
P20-250	1,57	1,57	1,57	1,53	1,53	1,53	0,57	0,57	0,57	0,14	0,14	0,14
P30-200	1,69	1,69	1,69	1,65	1,65	1,65	0,58	0,58	0,58	0,18	0,18	0,18
P40-150	1,73	1,73	1,73	1,71	1,71	1,71	0,58	0,58	0,58	0,19	0,19	0,19
P2-250	1,82	1,82	1,85	0	0	0	0,56	0,56	0,57	0,22	0,22	0,23
P3-200	1,72	1,75	1,75	0	0	0	0,56	0,56	0,56	0,23	0,23	0,23
P4-150	1,77	1,77	1,77	0	0	0	0,57	0,57	0,57	0,29	0,29	0,29
P5-100	2,06	2,08	2,08	0	0	0	0,60	0,60	0,60	0,43	0,44	0,44
P6-150	1,96	1,93	1,96	0	0	0	0	0	0	0,26	0,25	0,26
P7-120	2,08	2,08	2,10	0	0	0	0	0	0	0,32	0,32	0,33
P8-100	1,90	1,88	1,88	0	0	0	0	0	0	0,33	0,32	0,32
P11-100	1,45	1,48	1,45	2,18	2,27	2,18	0	0	0	0,11	0,12	0,11
P12-90	1,70	1,70	1,70	2,75	2,75	2,75	0	0	0	0,16	0,16	0,16
P13-80	1,49	1,49	1,49	2,34	2,34	2,34	0	0	0	0,13	0,13	0,13
P14-70	1,36	1,33	1,33	2,00	1,93	1,93	0	0	0	0,12	0,11	0,11
P21-70	1,88	1,76	1,76	0	0	0	0,45	0,45	0,45	0,07	0,06	0,06
P22-60	1,34	1,34	1,34	0	0	0	0,43	0,43	0,43	0,03	0,03	0,03
P23-50	2,56	2,66	2,56	0	0	0	0,48	0,49	0,48	0,14	0,15	0,14
P24-40	2,24	2,24	2,33	0	0	0	0,47	0,47	0,48	0,13	0,13	0,13
P41-100	0	0	0	4,00	4,00	3,70	0	0	0	0,05	0,05	0,05
P42-90	0	0	0	3,50	3,25	3,50	0	0	0	0,05	0,05	0,05
P43-80	0	0	0	3,38	3,60	3,60	0	0	0	0,06	0,06	0,06
PILH-300	0	0	0	1,20	1,20	1,20	0	0	0	0,02	0,02	0,02
PILH-250	0	0	0	1,39	1,39	1,39	0	0	0	0,03	0,03	0,03
PILH-200	0	0	0	1,30	1,30	1,30	0	0	0	0,03	0,03	0,03

Por fim, a próxima tabela (Tabela 12) exibe as datas limite (hora, no mês, em que os três lotes de peças devem estar prontos para despacho) utilizadas na simulação e que devem ter sido fornecidas quando do cadastro da peça, guia *Lote* (Figura 14, página 64). Considerou-se um conjunto de dados idêntico, para efeitos de comparação entre as heurísticas que requerem esta informação.

Tabela 12: Datas limites para os lotes.

Peça	Data limite
P1-300	121
P10-280	156
P20-250	106
P30-200	138
P40-150	154
P2-250	178
P3-200	114
P4-150	108
P5-100	133
P6-150	165
P7-120	115
P8-100	101
P11-100	152
P12-90	137
P13-80	125
P14-70	180
P21-70	97
P22-60	121
P23-50	167
P24-40	126
P41-100	154
P42-90	137
P43-80	151
PILH-300	142
PILH-250	100
PILH-200	117

Para os testes principais foram usadas, conforme comentado ao final do capítulo 4, as heurísticas *makespan* (utilização da planta), DLMA (data limite) e MTPP (fluxo), obtendo-se, no caso estático (isto é, $r_j = 0$ para qualquer j), os resultados que podem ser observados nas próximas figuras (Figuras 21-47). Nestas é possível identificar os processadores, as peças (rotuladas sobre

o bloco da carta de Gantt no formato "código da peça com 2 dígitos"–"número do lote com dois dígitos") e informações sobre o mês de execução, a heurística utilizada, o tempo total de utilização das máquinas, o custo de operação das mesmas e o resultado final.

Figura 21: *Makespan* em janeiro (1ª semana).

Figura 22: *Makespan* em janeiro (2ª semana).

Figura 23: *Makespan* em janeiro (3ª semana).

Figura 24: *Makespan* em março (1ª semana).

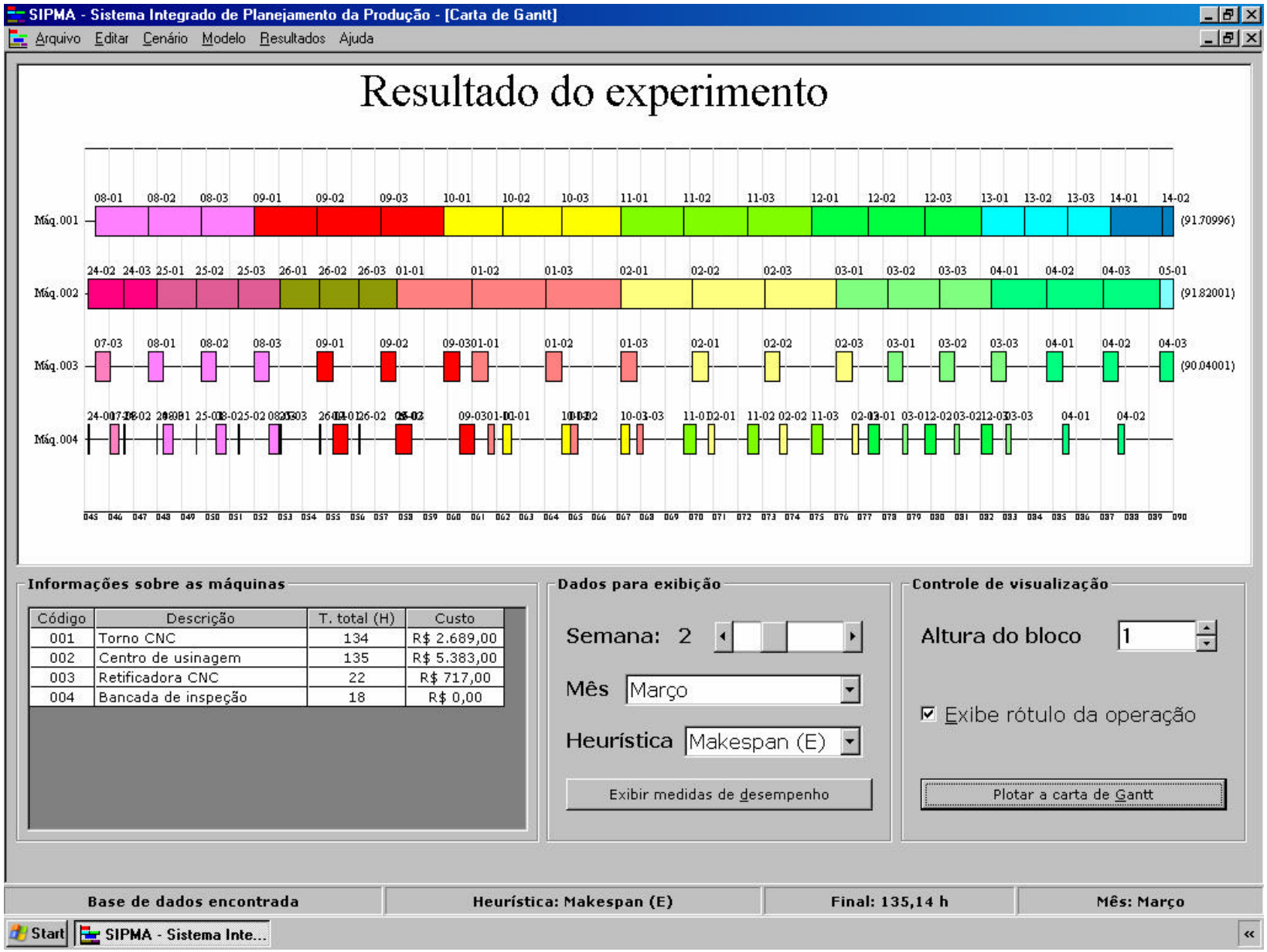


Figura 25: *Makespan* em março (2ª semana).

Figura 26: *Makespan* em março (3ª semana).

Figura 27: *Makespan* em dezembro (1ª semana).

Figura 28: *Makespan* em dezembro (2ª semana).

Figura 29: *Makespan* em dezembro (3ª semana).

Figura 30: *DLMA* em janeiro (1ª semana).

Figura 31: *DLMA* em janeiro (2ª semana).

Figura 32: *DLMA* em janeiro (3ª semana).

Figura 33: *DLMA* em março (1ª semana).

Figura 34: *DLMA* em março (2ª semana).

Figura 35: *DLMA* em março (3ª semana).

Figura 36: *DLMA* em março (4ª semana).

Figura 37: *DLMA* em dezembro (1ª semana).

Figura 38: *DLMA* em dezembro (2ª semana).

Figura 39: *DLMA* em dezembro (3ª semana).

Figura 40: *MTPP* em janeiro (1ª semana).

Figura 41: *MTPP* em janeiro (2ª semana).

Figura 42: *MTPP* em janeiro (3ª semana).

Figura 43: *MTPP* em março (1ª semana).

Figura 44: *MTPP* em março (2ª semana).

Figura 45: *MTPP* em março (3ª semana).

Figura 46: *MTPP* em dezembro (1ª semana).

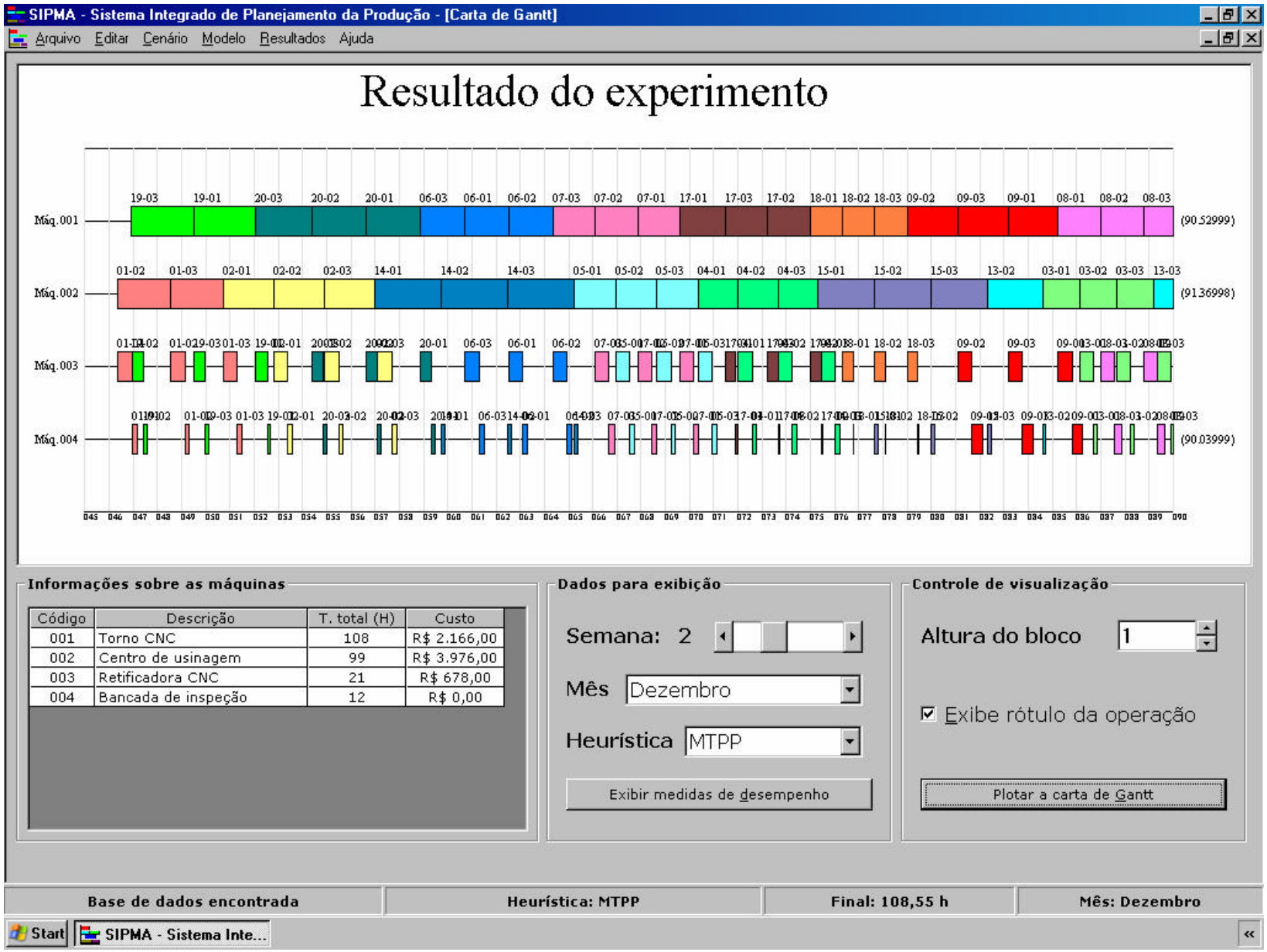


Figura 47: *MTPP* em dezembro (2ª semana).

Figura 48: *MTPP* em dezembro (3ª semana).

5.1 Discussão dos resultados

Da observação dos dados anteriormente apresentados são montadas as seguintes tabelas (Tabelas 13-15) que resumem as experimentações:

Tabela 13: Ordenação da produção segundo a heurística.

Ordem de entrada	Makespan	DLMA	MTPP (Jan)	MTPP (Mar)	MTPP (Dez)
1ª peça	1	17	1	21	1
2ª peça	2	25	21	4	2
3ª peça	3	12	2	13	14
4ª peça	4	3	5	22	5
5ª peça	5	8	22	3	4
6ª peça	6	7	15	23	21
7ª peça	7	11	4	16	15
8ª peça	8	26	13	19	13
9ª peça	9	1	3	20	3
10ª peça	10	18	23	1	23
11ª peça	11	15	16	14	22
12ª peça	12	20	19	2	16
13ª peça	13	9	20	17	19
14ª peça	14	22	14	5	20
15ª peça	15	14	17	15	6
16ª peça	16	4	18	18	7
17ª peça	17	24	25	25	17
18ª peça	18	23	8	26	18
19ª peça	19	13	26	8	9
20ª peça	20	21	7	6	25
21ª peça	21	5	11	7	26
22ª peça	22	2	10	24	8
23ª peça	23	10	9	10	24
24ª peça	24	19	12	9	11
25ª peça	25	6	6	11	12
26ª peça	26	16	24	12	10

Nos três meses de observação, as heurísticas *Makespan* e *DLMA* produziram a mesma seqüência de entrada das peças na planta. Entretanto a regra *MTPP* produziu em cada mês uma

seqüência diferente, pois a ordenação depende do peso w_j (fluxo), que variou devido à demanda em cada período (Proposições 9 e 10, páginas 46 e 47). A ordenação *DLMA* se manteve nos três meses de avaliação pois as datas-limite são as mesmas. No caso do *makespan*, a ordem foi definida pela soma dos tempos de processamento, sendo alocadas primeiro as tarefas em que tal soma fosse menor. Esta regra foi escolhida de modo arbitrário pois na ausência de ociosidade inserida qualquer ordenamento de atividades leva à solução ótima (Proposição 8, página 46).

Com relação aos tempos de término do seqüenciamento, tempos de uso das máquinas e custos associados, sejam as tabelas a seguir (Tabelas 14-16):

Tabela 14: Tempos de término dos seqüenciamentos.

Heurística / Mês	Janeiro	Março	Dezembro
Makespan	122,8	135,1	108,9
DLMA	126,5	139,3	111,6
MTPP	123,3	135,1	108,5
Variação (%)	± 3,0	± 3,1	± 2,9

Tabela 15: Tempo total de utilização de cada máquina (em horas).

Horas/Mês/Máquinas	Janeiro				Março				Dezembro			
	Torno	C. Us.	Ret.	Insp.	Torno	C. Us.	Ret.	Insp	Torno	C. Us.	Ret.	Insp
Makespan	122	118	22	15	134	135	22	18	108	99	21	12
DLMA	122	118	22	15	134	135	22	18	108	99	21	12
MTPP	122	118	22	15	134	135	22	18	108	99	21	12

Tabela 16: Custos totais de utilização de cada máquina (em unidades monetárias).

Horas/Mês/Máquinas	Janeiro				Março				Dezembro			
	Torno	C. Us.	Ret.	Insp.	Torno	C. Us.	Ret.	Insp	Torno	C. Us.	Ret.	Insp
Makespan	2445	4727	699	0	2689	5383	717	0	2166	3976	678	0
DLMA	2445	4727	699	0	2689	5383	717	0	2166	3976	678	0
MTPP	2445	4727	699	0	2689	5383	717	0	2166	3976	678	0

O percentual descrito na Tabela 14 se refere à variação do tempos de encerramento dos seqüenciamentos fornecidos pelas três heurísticas selecionadas para compor esta análise. É possível observar que no mês de dezembro (em que houve menor demanda por peças e, portanto, a planta foi menos utilizada) ocorreu a menor variação, o que pode indicar uma possível equivalência entre as regras adotadas para utilizações não-intensivas do sistema. O conceito de "não-intensividade", claro está, é variável de acordo com as circunstâncias, como número de máquinas, processos e os roteiros de fabricação.

Como pode ser notado, os resultados de março (mês de maior atividade) e dezembro (menor atividade) para MTPP e *makespan* são próximos; isto é motivado pela semelhança de propósito entre as duas funções: a primeira busca minimizar o fluxo de atividades na planta e a segunda trata de otimizar a utilização desta o que, de certa forma, significa, igualmente, tornar o fluxo menor.

Do mesmo modo, em todos os meses, percebe-se o maior resultado fornecido pela função DLMA. Isto pode significar um comprometimento maior do uso da planta para que se obedeçam, o mais possível, as datas-limite. Como o uso das máquinas teve o mesmo perfil e, por conseguinte, implicará em mesmo custo (Tabela 16), conclui-se ser a política DLMA preferível apenas em casos cujas datas-limite devam ser *estritamente* seguidas.

A produção em lotes permitiu maior flexibilidade na alocação das atividades. Por causa da fórmula de *setup* rateado a produção de apenas um lote ou de lotes com diferença significativa de tamanho poderia implicar em ultrapassar as capacidades diárias de cada processador. Para a compreensão deste fato, seja considerada a tabela abaixo (Tabela 17):

Tabela 17: Exemplo de tempos de processamento para a peça P1-300.

Perfil de utilização do torno para a peça P1-300 (tempos em minutos)										
L1	**L2**	**L3**	***Setup***	**TP**	**1 pcs (L1)**	**1 pcs (L2)**	**1 pcs (L3)**	**Total (L1)**	**Total (L2)**	**Total (L3)**
85	0	0	55	2,67	3,3	57,7	57,7	287,29	0	0
29	29	29	55	2,67	4,6	4,6	4,6	132,43	132,43	132,43

As primeiras três colunas (**L1**, **L2** e **L3**) apresentam duas situações: uma em que o primeiro lote é a totalidade das peças a serem processadas (total de 87) e outra em que os três lotes são igualmente distribuídos. As colunas ***Setup*** e **TP** apresentam, respectivamente, o tempo de *setup* do torno para processar P1-300 e seu tempo de processamento. As três colunas seguintes da Tabela 15 mostram o tempo para produzir 1 peça, por meio da fórmula de *setup* rateado (Equação [32], página 67) e as três colunas finais mostram o tempo total de processamento de cada lote. Percebe-se claramente que a produção dos três lotes igualmente distribuídos demanda um tempo substancialmente maior do que a produção do lote em apenas uma parcela de 87 peças (398 minutos aproximadamente contra 287 minutos). Entretanto, havendo mais de um lote, estes podem ser rearranjados no tempo e entre um e outro é possível ocorrer o processamento de outra peça naquele torno o que, em termos da política DLMA, pode ser útil, dado o cumprimento das datas-limite, mas pode, contrariamente, influenciar negativamente na política MTPP, pois haverá maior fluxo de atividades segmentadas.

Pela observação das cartas de Gantt (Figuras 21-48) nota-se a utilização praticamente ininterrupta do torno CNC (Máq. 001) e do Centro de Usinagem CNC (Máq. 002), visto trabalharem peças com tempos totais de processamento relativamente maiores em relação aos mesmos tempos nas operações ocorridas na retificadora e na bancada de inspeção; nestes locais a taxa de ocupação fica, portanto, condicionada à seqüência de produção determinada pelas heurísticas. Por exemplo, a utilização da máquina retificadora (Máq. 003) pela ordenação DLMA é regular nas três semanas de experimentação, ao passo que pela ordenção via *makespan* esta utilização se concentra mais do meio da primeira semana em diante e pela ordenação MTPP ocorre quase que exclusivamente na segunda semana nos meses de dezembro e janeiro e na segunda e terceira semanas no mês de março. Na bancada de inspeção (Máq. 004) o que se observa é que para as ordenações por *makespan* e MTPP sua ocupação crítica se dá, nos três meses de estudo, nas duas últimas semanas e na ordenação DLMA ocorre uma ocupação regular, como no caso da retificadora. Tais informações podem ser importantes no planejamento futuro de peças que se utilizem apenas destas duas máquina (i.e. que já tenham sido torneadas e / ou processadas num centro de usinagem em outra célula) e que entram dinamicamente no sistema; deste modo será possível ajustar seus processamentos entre os tempos ociosos de ambos os locais.

5.1.1 Medidas de desempenho e funções-objetivo

Com base nas medidas de desempenho enunciadas no capítulo 3 (às páginas 35 e 36) e que são calculadas automaticamente pelo SIPMA quando do término do seqüenciamento, seja a tabela (Tabela 18) a seguir, que exibe os valores das funções-objetivo definidas em termos de tais medidas (páginas 37-39)

Tabela 18: Valores das funções-objetivo para cada heurística.

		C_{max}	F_{wt}	L_{wt}	T_{wt}	N_{wt}	ET_{wt}	F_{max}	L_{max}	T_{max}
Janeiro	*Makespan*	**122,82**	**3465,76**	**-3551,24**	**5,66**	**1,5**	**3562,56**	**122,82**	**5,64**	**5,64**
	DLMA	**126,47**	**3070,2**	**-3946,8**	**0**	**0**	**3946,8**	**126,47**	**-43,88**	**0**
	MTPP	**123,31**	**4352,91**	**-2664,09**	**29,38**	**4**	**2772,85**	**123,31**	**9,6**	**9,6**
Março	*Makespan*	**135,14**	**3861,77**	**-3155,24**	**30,27**	**3,5**	**3215,76**	**135,14**	**15,52**	**15,52**
	DLMA	**139,3**	**3417,21**	**-3599,79**	**0**	**0**	**3599,79**	**139,3**	**-28,39**	**0**
	MTPP	**135,13**	**4646,13**	**-2370,88**	**135,62**	**7,5**	**2642,11**	**135,13**	**33,93**	**33,93**
Dezembro	*Makespan*	**108,91**	**3009,92**	**-4007,09**	**0**	**0**	**4007,09**	**108,91**	**-5,8**	**0**
	DLMA	**111,61**	**2675,08**	**-4341,92**	**0**	**0**	**4341,92**	**111,61**	**-56,5**	**0**
	MTPP	**108,55**	**3696,61**	**-3320,39**	**9,5**	**2**	**3339,39**	**108,55**	**5,69**	**5,69**

O atendimento aos pedidos é sempre o fator de fundamental importância na determinação da heurística que estabelecerá o seqüenciamento da produção, que também está relacionado ao nível de utilização da planta. Portanto, define-se como a política de escolha da melhor regra aquela que se ajustar, o máximo possível:

- Ao uso da planta (baixa utilização, padrão ou alta utilização) e
- À satisfação do cliente (aceitação de entregas com atraso ou com adiantamento, insatisfação progressiva).

A Tabela 19 exibe um panorama que auxilia a escolher a regra que melhor se adapta aos fatores acima listados pela observação das funções-objetivo da Tabela 18:

Tabela 19: Melhores regras para a satisfação do cliente.

		Satisfação		
		FS1	FS2	FS3
Utilização	Baixa	MTPP	DLMA	DLMA
	Padrão	MTPP	DLMA	DLMA
	Alta	MTPP	*Makespan* DLMA	*Makespan* DLMA

Na tabela acima, FS1, FS2 e FS3 representam o *Fator de Satisfação* do cliente, a saber:

- FS1: O cliente tolera atrasos mas não aceita a encomenda antes do prazo (função ET_{wt});
- FS2: O cliente não aceita atrasos (Função N_{wt});
- FS3: O cliente tolera atrasos e sua insatisfação é progressiva caso o atraso se prolongue (função T_{max}, equivalente à função L_{max}).

A definição do que é utilização "baixa", "padrão" e "alta" para este exemplo baseou-se no perfil de demanda (Tabela 5, página 89) de modo a coincidir com os meses em estudo. Assim, a *utilização padrão* é aquela que se verificou na produção de peças em quantidades iguais ao valor base definido (janeiro); a *baixa utilização*, por sua vez, foi a observada no mês de dezembro, em que se produziu 20% abaixo do valor base e a *alta utilização* foi em março, em que se produziu 15% acima do valor base.

Quanto às funções de fluxo, o cumprimento da condição FS1 levou também à minimização deste nos meses de utilização alta e baixa, como se pode observar na Tabela 18; entretanto, o cumprimento de FS2 e FS3, contrariamente, implica em abrir mão desta minimização nos períodos de utilização padrão e baixa. No período de alta utilização, para F2 e F3, é preferível optar pela regra do *makespan* que, embora não forneça uma solução ótima, oferece uma alternativa para tornar o fluxo na planta menor.

Conclui-se, portanto, que as heurísticas baseadas em fluxo (*makespan* e MTPP) apresentaram os melhores resultados em ocasião de utilização mais intensa da planta e a

heurística baseada em data limite (DLMA) solucionou melhor o problema em situações de utilização baixa.

5.1.2 Tempo computacional de execução

Com relação ao tempo computacional, verificou-se a rápida execução dos modelos, por se tratar apenas de ordenação das atividades baseada nas regras selecionadas. A colocação das tarefas na ordem imposta pelas políticas de *makespan*, DLMA e MTPP seguiu a ordenação tipo bolha (*Bubble sort* – Knuth, 1998). O SIPMA foi inicialmente testado numa configuração baseada no processador Intel® Pentium II® a 266 MHz e 64 MB de memória RAM, com o sistema operacional Microsoft® Windows 98™, recurso disponível no LMA (Laboratório de Manufatura Assistida), DEF – FEM – UNICAMP, tendo sido computados os seguintes tempos para a obtenção dos resultados (Tabela 20):

Tabela 20: Tempos de execução do SIPMA para o cálculo dos resultados.

Tempos de execução (em segundos)			
Heurística / Mês	**Janeiro**	**Março**	**Dezembro**
Makespan	30	30	31
DLMA	12	15	15
MTPP	29	35	32

De um modo geral foi possível determinar todos os seqüenciamentos para que não excedessem o horizonte de tempo, estipulado em 180 horas / mês.

Capítulo 6

Conclusões e sugestões para próximos trabalhos

A importância da programação de atividades tem feito com que cada vez mais fossem estudadas as propriedades deste importante problema de otimização. Principalmente a partir do momento em que a competitividade entre indústrias começou a impor um novo paradigma de produção, a programação eficiente de atividades em chão de fábrica conquistou, definitivamente, um espaço importante no planejamento de uma firma.

O presente texto buscou apresentar algumas características típicas de determinados problemas de programação, incluindo as aproximações matemáticas exatas, heurísticas e mistas. Tal caracterização justificou a elaboração de um sistema que permitisse coordenar os trabalhos de programação, representado pelo SIPMA (Sistema de Planejamento de Manufatura e Automação). Dentre as abordagens possíveis, escolheu-se para discussão e formalização teórica as heurísticas, por sua importância em termos de facilidade de implementação e cenários que podem ser prontamente avaliados, uma vez que as regras de decisão têm sua origem direta na observação e histórico de programações anteriores, o que pode levar a métodos eficientes para determinadas situações.

No que se refere ao desenvolvimento de sistemas computacionais que implementam heurísticas de programação, diversas soluções foram apresentadas, inicialmente com propósitos acadêmicos, posteriormente passando a constituir uma classe de ferramentas cujo uso atual é o diferencial competitivo de diversas indústrias.

A programação de heurísticas exibe uma considerável diferença de tempo no que se refere à resolução. Por impor regras simples de sequenciamento e temporização, apenas estas são as decisões que devem ser tomadas e, assim, o tempo de processamento para a obtenção de um resultado é consideravelmente menor em relação aos métodos exatos que, em sua maioria, buscam respostas enumerando, ainda que de modo controlado, todas as possíveis soluções. Com relação a precisão entre uma e outra abordagem, a exatidão conseguida pela programação matemática supera a observada pelo uso de regras de decisão; entretanto esta mesma exatidão (*quando conseguida*) é um fator que torna proibitivo o emprego de tais técnicas dado o tempo computacional de execução de um problema assim formulado. Além disso, o bom senso e a observação do comportamento da planta ao longo do tempo são fatores que tornam as heurísticas o método preferencial quando se trata de programação de plantas complexas, substituindo a exatidão pelo conhecimento inerente advindo do estudo de seu histórico, sem dúvida mais revelador de seus detalhes.

O SIPMA, em sua versão inicial, tem a capacidade de programar atividades em regime estático ou dinâmico, permitindo ou não preemptividade. Entretanto, não foi considerado um ponto importante, que se refere aos tempos de *setup* dependentes da seqüência. Como sugestão à futuras melhorias no sistema, pode-se, portanto, estabelecer a implementação de tais rotinas, permitindo maior flexibilidade ao usuário. Apesar de eficiente, o SIPMA foi implementado de modo a cobrir o problema particular proposto como teste (ver capítulo 4). Uma nova sugestão é a cobertura de uma extensão maior de problemas, permitindo a partição de lotes totais em número diferente de três, bem como permitir o uso de mais processadores.

O leque de possibilidades de implementação e melhorias do SIPMA é muito grande, e pretende-se introduzir constantes aperfeiçoamentos neste *software*, tornando-o um sistema profissional e possibilitando a maior integração entre o desenvolvimento acadêmico e o setor industrial.

Referências bibliográficas

Aarts, E., Van Laarhoven, P. *et al.*, *Simulated annealing*, em E. *Aarts and J.K. Lenstra* (editores) Local Search in Combinatorial Optimization, John Wiley & Sons Ltd., 1997

Agostinho, O. L., *Sistemas de manufatura – volume I*, apostila do curso de IM191, Universidade Estadual de Campinas (UNICAMP), 1997

Baker, K. R., *Introduction to sequencing and scheduling*, John Wiley & Sons, 1974 - livro

Bazaraa, M. S., *Linear Programming and Network Flows*, John Wiley & Sons, 1990, pp. 278

Burbidge, J. L., *Introduction to group technology*, William Heinemann Ltd., 1975 - livro

Carroll, D. C., *Heuristic sequencing of single and multiple component jobs*, Sloan School of Management, M.I.T., Cambridge, 1965 - livro

Coffman, E. G. *et al.*, *Computer and job-shop scheduling theory*, John Wiley and Sons, 1976

Dasgupta, D. *et al.*, *Designing Application-Specific Neural Networks using the Structured Genetic Algorithm*, em *Proceedings of COGANN-92*, D. Whitley e J.D. Schaffer (Editores), IEEE, Computer Society Press. E.U.A., 1992

Durkin, J., *Expert Systems: Design and Development*, Prentice Hall, Englewood Cliffs, 1994 - livro

Edquist, C. *et al.*, *Flexible automation: the global diffusion of new engineering technology*, Basil Blackwell, 1988 – livro

Glover, F., Laguna, M. *et al.*, *Tabu Search*, Annals of Operations Research, volume 41, 1993

Hillier, F.S. *et al.*: *Introduction to Operations Research*, Holden-Day, 1967 (Holden-Day series in industrial engineering and management science) - livro

Jackson, D., *Cell systems of production – an effective organisation structure*, Business Books, 1978 - livro

Johnson, L. A. *et al.*, *Operations research in production planning, scheduling and inventory control*, John Wiley & Sons, 1974 - livro

Judson, D. L., Integrated manufacturing control material management (IMC-MM), documento técnico da Society of Manufacturing Engineers, MS79-841, Dearborn, Michigan, 1979

Kinney Jr., H. D. *et al.*, *Design and control of manufacturing cells,* Facilites & Material Systems Series part 7, Material Handling Systems, 1987 (a)

Kinney Jr., H. D. *et al.*, *Manufacturing cells solve material handling problems*, Facilites & Material Systems Series part 7, Material Handling Systems, 1987 (b)

Knuth, D., *The Art of Computer Programming: Sorting and Searching*, 2ª edição, vol. 3. Addison-Wesley, 1998. - livro

Kondili, E. *et al.*, *A General Algorithm for Short-term Scheduling of Batch Operations – I. MILP Formulation*, Computers chem. Engng., Vol. 17, No.2, pp. 211-227, 1993, Pergamon Press

Kusiak, A., *Intelligent manufacturing systems*, Prentice Hall International Series in Industrial and Systems Engineering, 1990 - livro

Morton, T. E. *et al.*, *Heuristic scheduling systems with applications to production systems and project management*, Wiley series in engineering & techonology managemente, John Wiley & Sons, 1993 - livro

Nahmias, S., *Production and Operation Analysis*, Irwin, 1993 - livro

Narasimhan, S. L. *et al.*, *Production planningand inventory control*, Quantitative Methods and Applied Statistics series, Prentice-Hall, 1995 - livro

Papadopoulos, H. T. *et al.*, *Queuing theory in manufacturing systems analysis and design*, Chapman & Hall, 1993 - livro

Parsaei, H. R. *et al.* (editores), *Justification methods for computer integrated manufacturing systems*, Manufacturing Research and Technology Series / 9, Elsevier, 1990 - livro

Pinedo, M., *Scheduling theory, algorithms and systems*, Prentice-Hall, 1995 - livro

Ránky, P., *The design and operation of FMS,* IFS (Publications) Ltd, UK, North-Holland Publishing Company, 1983 - livro

Rathmill, K. (editor), *Proceedings of the 2nd international conference of flexible manufacturing systems*, 26-28 outubro de 1983, Londres, IFS (Publications) Ltd, UK, North-Holland Publishing Company

Rembold, U. *et al.*, *Computer-integrated manufacturing technology and systems*, Manufacturing Engineering and Materials Processing Series / 16, Marcel-Dekker, Inc., 1985 - livro

Rich, S. H. *et al.*: *Scheduling and Sequencing of Batch Operations in a Multipurpose Plant*, Ind. Eng. Chem. Res., 25(4), 1986

Skinner, W., *The taming of lions: how manufacturing leadership evolved, 1780-1984*, capítulo 2 em *The uneasy alliance: managing the productivity-technology dilemma*, K. B. Clark, R. H. Hayes e C. Lorenz, Boston: Harvard Business School, 1985 - livro

Zhang, W., *Complete anytime beam search,* em *Proceedings of AAAI-98*, Madison, WI, 1998

Anexo I

Estrutura das *strings* utilizadas pelo SIPMA

1) Formato da string:

QQCM1PL1CM2PL2CM3PL3...

Dados:

QQ = quantidade de máquinas que podem executar aquela operação (2 dígitos)
CM1 = Máquina 1 (código); PL1 = Uso interno do SIPMA (ambos com 3 dígitos)
CM2 = Máquina 2 (código); PL2 = Uso interno do SIPMA (ambos com 3 dígitos)
CM3 = Máquina 3 (código); PL3 = Uso interno do SIPMA (ambos com 3 dígitos)...

Em que campo é armazenada: Opr_MAQ **/** Tabela **Operações**

Finalidade: armazenar informações sobre quantas e quais máquinas podem desempenhar a operação em edição. *PLx* é um indicador interno que garante que as máquinas serão exibidas na ordem correta quando de sua atribuição, ao se visualizar a peça no modo de edição.

2) Formato da string:

0000IGU, NNNNACI ou NNNNABA x 12 (um para cada mês)

Dados:

NNNN = Percentual (inteiro com 4 dígitos)
IGU = Igual **/** ACI = Acima **/** ABA = Abaixo

Em que campo é armazenada: Pcs_PDM **/** Tabela **Peças**

Finalidade: armazenar informações sobre o perfil anual da demanda para cada peça (isto é, o percentual acima ou abaixo do valor base definido na guia *Demanda*; 0000IGU indica que naquele mês a demanda é igual ao valor base).

3) Formato da string:

BBBB + MMVVVV x 12 (um para cada mês)

Dados:

BBBB = Valor base da demanda para a peça em edição (inteiro com 4 dígitos)

MM = Mês (inteiro com dois dígitos)

VVVV = Valor da demanda naquele mês, segundo o perfil calculado anteriormente

Em que campo é armazenada: Pcs_DEM / Tabela **Peças**

Finalidade: armazenar informações sobre a demanda mensal de cada peça. O sinal de "+" no formato da *string* indica que o valor "BBBB" entrará apenas uma vez na composição da mesma, ocupando sempre as quatro primeiras posições. Depois, o padrão "MMVVVV" será repetido 12 vezes, um para cada mês.

4) Formato da string:

(MM + 111111222222333333444444555555666666777777 (x 3)) x 12 (um para cada mês)

Dados:

MM = Mês (inteiro com dois dígitos)

111111 = Tamanho do lote (inteiro com 6 dígitos)

222222 = Início do processamento (r_j) (inteiro com 6 dígitos)

333333 = Peso por fluxo (real com 6 dígitos incluindo a vírgula)

444444 = Peso por adiamento (real com 6 dígitos incluindo a vírgula)

555555 = Peso por adiantamento (real com 6 dígitos incluindo a vírgula)

666666 = Peso por número de atividades adiadas (real com 6 dígitos incluindo a vírgula)

777777 = Data limite de entrega (real com 6 dígitos incluindo a vírgula)

Em que campo é armazenada: Pcs_LOT / Tabela **Peças**

Finalidade: armazenar informações sobre os detalhes do lote peças. O sinal de "+" no formato da *string* indica que o valor "MM" entrará apenas uma vez na composição de cada um dos 12 grupos, ocupando sempre as duas primeiras posições. Depois, o padrão "111111222222333333444444555555666666777777" será repetido 3 vezes, um para cada lote.

5) Formato da string:

QQCM1CM2CM3...

Dados:

QQ = Quantidade de operações pelas quais a peça passa (2 dígitos)

CM1 = Máquina 1 (código) (3 dígitos)

CM2 = Máquina 2 (código) (3 dígitos)

CM3 = Máquina 3 (código) (3 dígitos)

Em que campo é armazenada: Pcs_ROT **/** Tabela **Peças**

Finalidade: armazenar informações sobre as máquinas que processarão as peças em seu ciclo de produção. Nesta *string* o posicionamento de seus códigos indica exatamente a ordem das mesmas pelas quais a peça passará, diferentemente do campo **Opr_MAQ**, em que a ordem não importava.

6) Formato da string:

(OPXXOOOPQQQ+((MM1X1X1X1Y1Y1Y1) x QQQ)) x QQ da *string* (5) anterior

Dados:

OPXX = Identificação da operação (por exemplo, OP01, OP02, OP03...) (4 dígitos)

OOO = Código desta operação na tabela **Operações** (3 dígitos)

P = Política de ocupação (1 dígito)

QQQ = Quantas máquinas podem desempenhar esta operação (inteiro com 3 dígitos)

MM1 = Código da primeira máquina (a principal) que pode desempenhar esta operação (3 díg.)

X1X1X1 = Tempo de *setup* desta máquina (6 dígitos, incluindo a vírgula)

Y1Y1Y1 = Tempo de processamento da peça nesta máquina (6 dígitos, incluindo a vírgula)

Em que campo é armazenada: Pcs_DRO **/** Tabela **Peças**

Finalidade: armazenar informações detalhadas sobre o roteiro de fabricação da peça. Para cada operação, os 11 primeiros caracteres (OPXXOOOPQQQ) são fixos e haverá *QQQ* repetições do

conjunto (MM1X1X1X1Y1Y1Y1), cada um indicando a máquina, seu tempo de *setup* e o tempo de processamento da peça nesta máquina. A primeira é a máquina principal e as demais, se existirem, serão as alternativas, a serem ocupadas ou não dependendo da política de ocupação (P) escolhida. Esta *string* será repetida *QQ* vezes, em que QQ é a quantidade de operações pelas quais a peça passa (2 dígitos) em seu roteiro de fabricação, definida na *string* (5).

7) Formato da string:

XX + (OPXXOOOPQQQ+((MM1X1X1X1) x QQQ)) x QQ da *string* (5) anterior

Dados:

XX = Quantidade de operações (inteiro com 2 dígitos)

OPXX = Identificação da operação (por exemplo, OP01, OP02, OP03...) (4 dígitos)

OOO = Código desta operação na tabela **Operações** (3 dígitos)

P = Política de ocupação (1 dígito)

QQQ = Quantas máquinas podem desempenhar esta operação (inteiro com 3 dígitos)

MM1 = Código da primeira máquina (a principal) que pode desempenhar esta operação (3 díg.)

X1X1X1 = Tempo total de processamento da peça naquela máquina (6 dígitos, incluindo a vírgula)

Em que campo é armazenada: Cen_OPT / Tabela **Cenário**

Finalidade: armazenar informações detalhadas sobre o roteiro de fabricação da peça em termos de seus tempos finais de processamento. Possui, praticamente, a mesma estrutura a *string* (6).

Exemplos das strings acima definidas:

String (1)

01001001

String (2)

000IGU010ACI015ACI020ABA000IGU010ABA010ACI015ACI010ABA010ACI010ACI020ABA

String (3)

00870100870200960301020400700500870600780700960801000900781000961100961200 69

String (4)

0100002900000000 0,50000,50000,50000,5000012100002900000000 0,50000,50000,50000,5000012100002900000000,50000,50000,50000,5000012102000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000003000034000000001,00001,00001,00001,0000012100003400000000 1,00001,00001,00001,000001210000340000000 1,00001,00001,00001,000001210400000000 0000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000005000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,00000000000000000000,00000,00000,00000,000000006000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000000000000000000,00000,00000,00000,000000000700000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000000800000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000000000000000000,00000,00000,00000,000000009000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000010000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000001100000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000000000000000000000,00000,00000,00000,000000001200002300000000 0,50000,50000,50000,5000012100002300000000 0,50000,50000,50000,5000012100002300000000 0,50000,50000,50000,50000121

String (5)

04001002003004

String (6)

OP010010001001055,00002,67OP020020001002015,00005,00OP030030001003030,00000,27OP040040001004000,00000,50

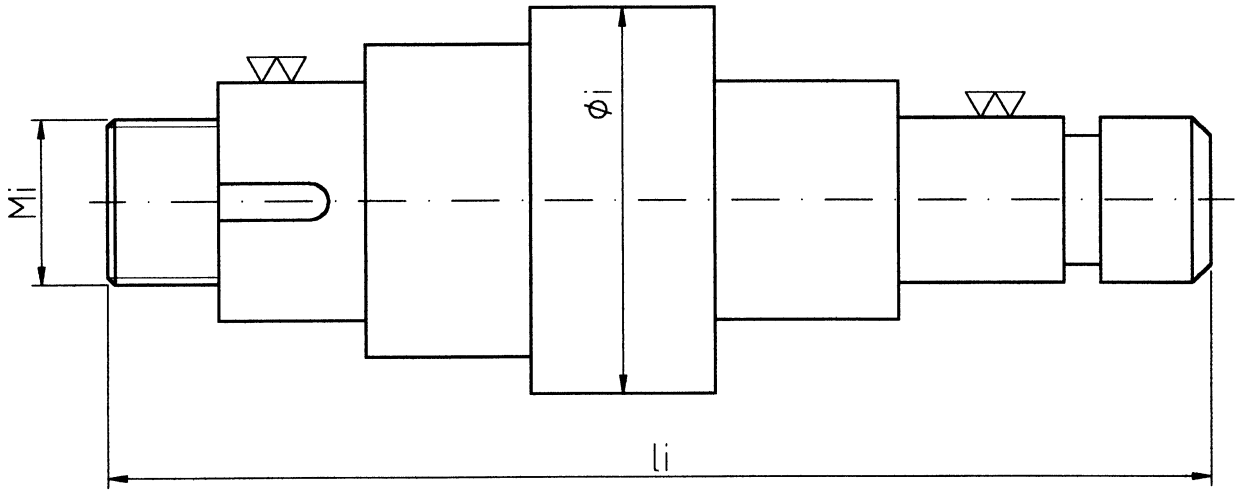
String (7)

004OP010010001001002,21OP020020001002002,67OP030030001003000,63OP040040001004000,24

Anexo II

Famílias de peças criadas para o teste do SIPMA

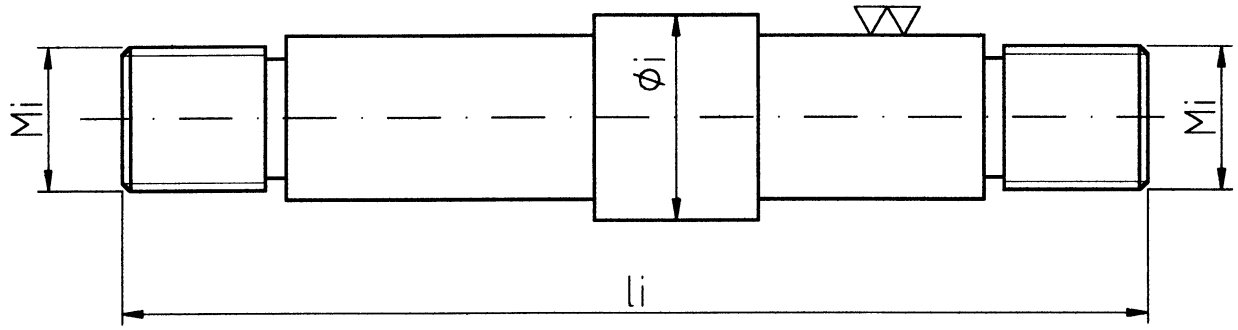
Família E1:



Dimensões

	l_i	ϕ_i	M_i
P1-300	300	70	M45
P10-280	280	50	M40
P20-250	250	50	M35
P30-200	200	50	M30
P40-150	150	50	M30

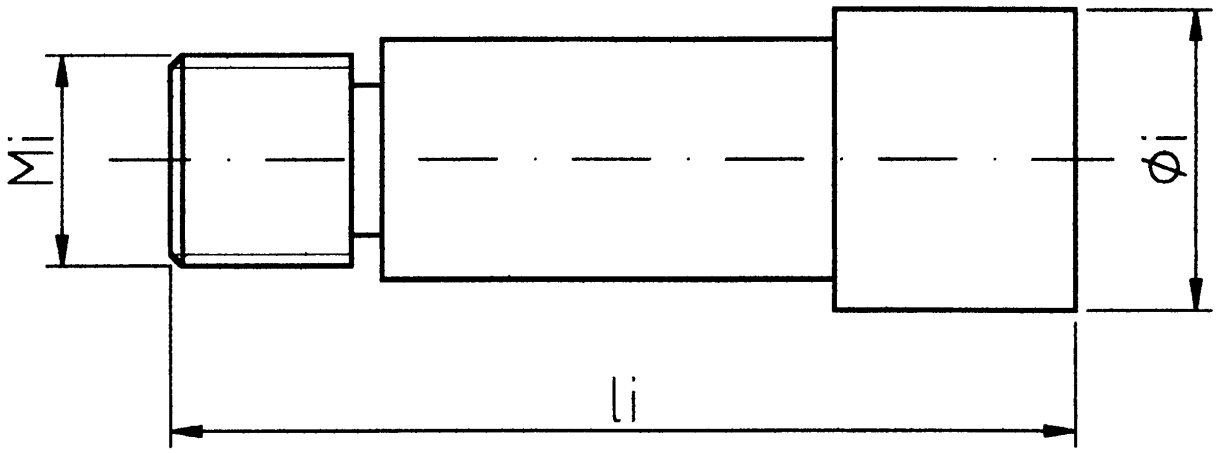
Família E2:



Dimensões

	l_i	ϕ_i	M_i
P2-250	250	50	M35
P3-200	200	45	M30
P4-150	150	40	M20
P5-100	100	35	M20

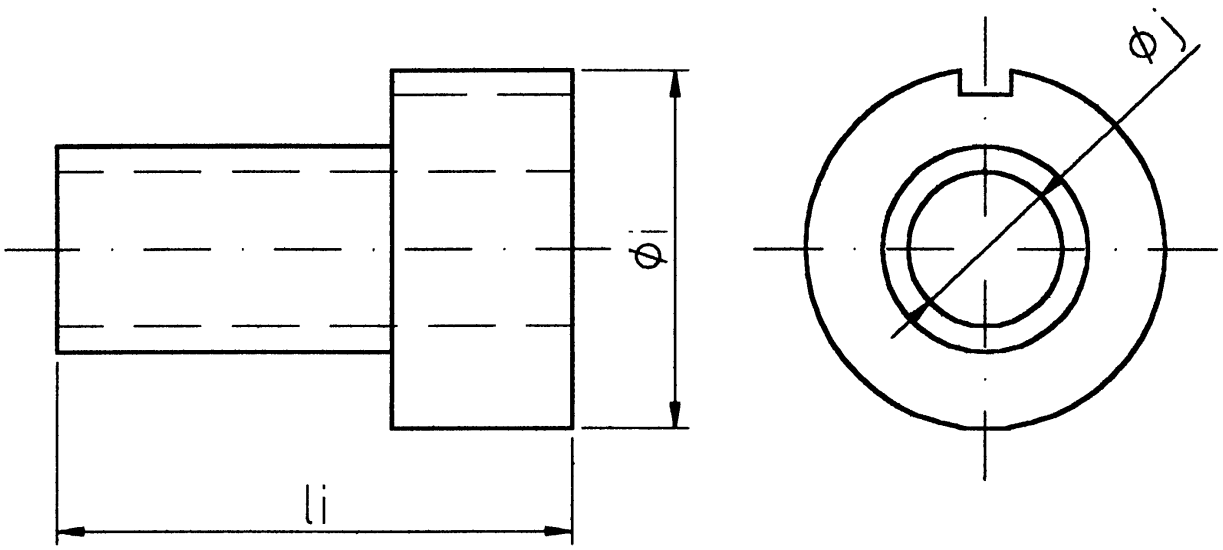
Família E3:



Dimensões

	l_i	ϕ_i	M_i
P6-150	150	50	M35
P7-120	120	40	M25
P8-100	100	30	M20

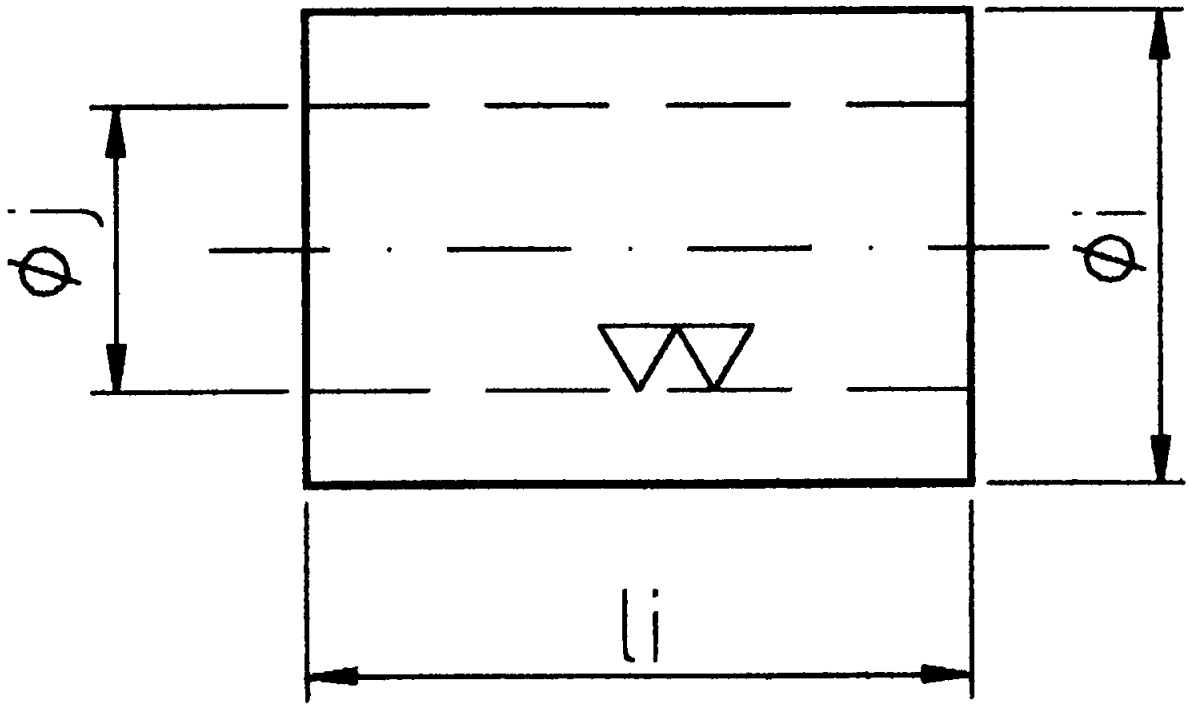
Família B1:



Dimensões:

	l_i	ϕ_i	ϕ_j
P11-100	100	70	35
P12-90	90	60	30
P13-80	80	60	30
P14-70	70	50	20

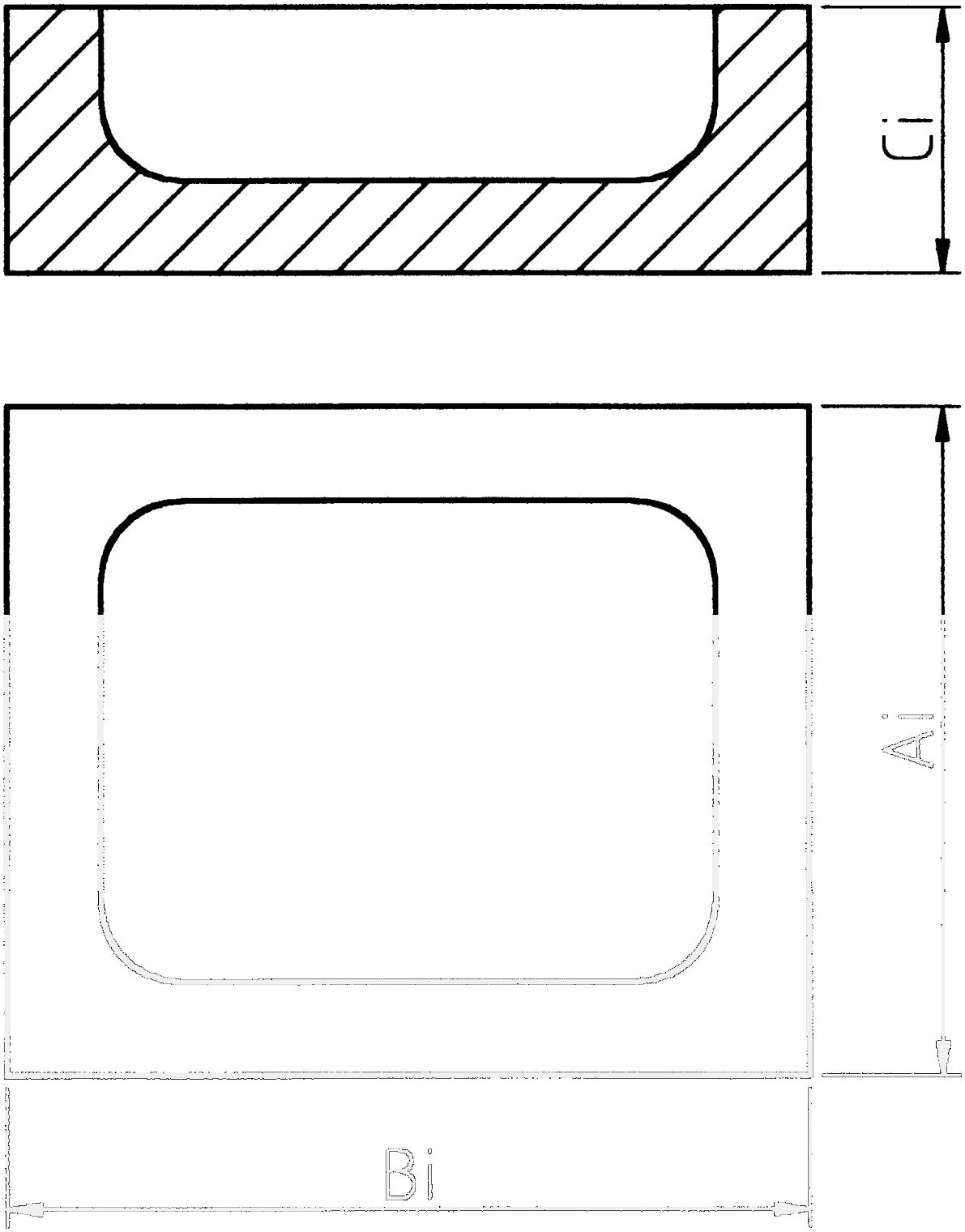
Família B2:



Dimensões:

	l_i	ϕ_i	ϕ_j
P21-70	70	50	30
P22-60	60	50	30
P23-50	50	40	25
P24-40	40	30	25

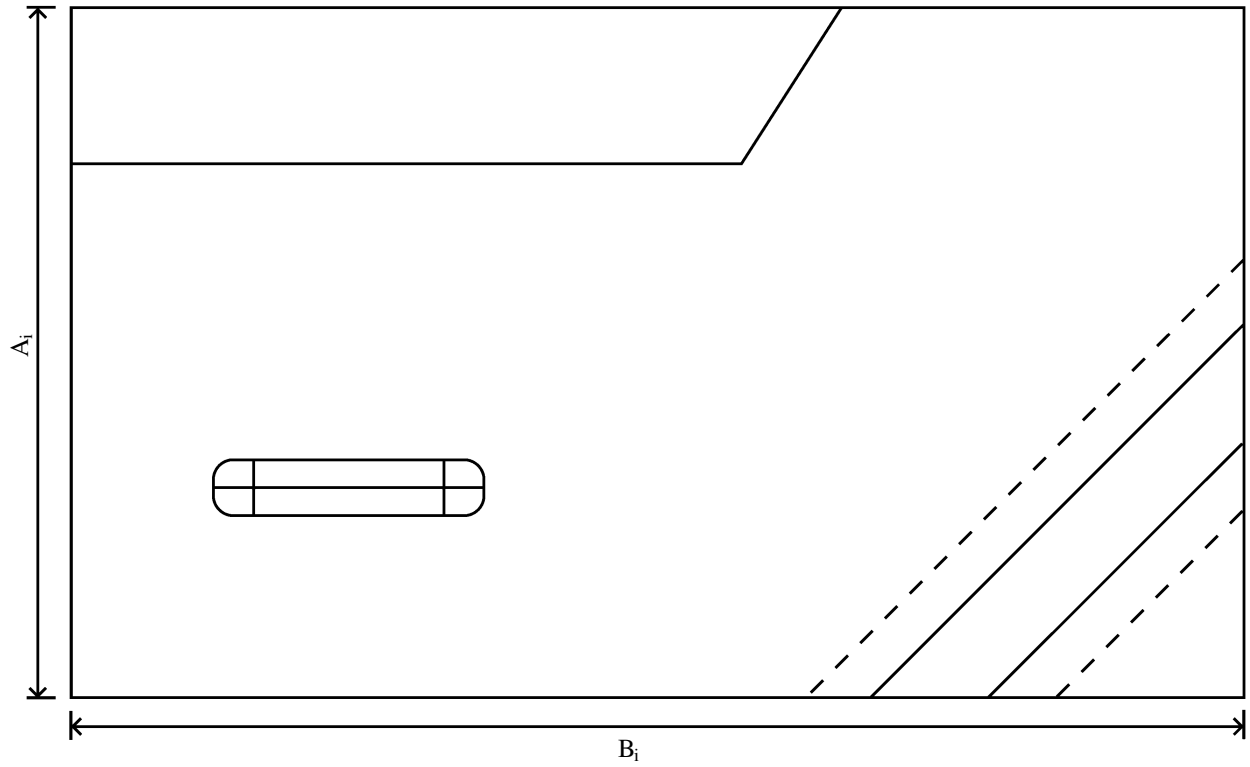
Família S1:



Para a peça da página anterior, as medidas são

	A_i	B_i	C_i
PILH-300	300	200	40
PILH-250	250	200	35
PILH-200	200	200	35

Família S2:



Dimensões

	A_i	B_i
P41-100	120	180
P42-90	130	170
P43-80	140	160