UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE PRODUÇÃO E SISTEMAS

Uma Contribuição para o Escalonamento da Produção baseado em Métodos Globalmente Distribuídos

Maria Leonilde Rocha Varela

Tese de Doutoramento
2007

Uma Contribuição para o Escalonamento da Produção baseado em Métodos Globalmente Distribuídos

Maria Leonilde Rocha Varela

Tese submetida para satisfação parcial dos requisitos para a obtenção do grau de Doutor em Engenharia de Produção e Sistemas

Orientação científica de
Sílvio do Carmo Silva

Universidade do Minho
Julho de 2007

Dedico este trabalho aos meus pais,
Domingos e Graça.

Agradecimentos

Quero aqui expressar os meus agradecimentos a todos os que, de um modo directo ou indirecto, contribuíram para a realização deste trabalho.

À Universidade do Minho, pela oportunidade de realização desta tese.

Ao Departamento de Produção e Sistemas, pela motivação, compreensão e tolerância prestadas ao longo do decurso desta tese.

Ao Dr. Aparício, meu professor no mestrado em Inteligência Artificial Aplicada, da Universidade Nova de Lisboa, que infelizmente já não se encontra entre nós, pelo seus ensinamentos importantes ao desenvolvimento do sistema web relatado nesta tese.

E, finalmente ao meu orientador científico, Dr. Sílvio do Carmo Silva, pela sua enriquecedora orientação e pelo acompanhamento e ajuda incessante ao longo do desenvolvimento deste trabalho.

Resumo

O escalonamento da produção é uma função que pode contribuir fortemente para a capacidade competitiva das empresas produtores de bens e serviços. Pode, simplificadamente ser definido com a função de afectar tarefas a meios de produção ao longo do tempo. Resolver um problema de escalonamento envolve várias actividades. Primeiro é necessário descrever o problema e identificá-lo numa dada classe. Depois, é necessário encontrar um método eficiente apropriado à sua resolução e usá-lo para resolver o problema, disponibilizando os dados necessários e tratando os resultados de forma conveniente.

Muitas vezes as pessoas envolvidas na resolução de problemas de escalonamento não conhecem os métodos ou não sabem como ter acesso a eles, ou mesmo não sabem como formalmente identificar o problema numa classe.

O trabalho de investigação aqui relatado pretende contribuir para facilitar a compreensão dos problemas e melhorar o processo de escalonamento na indústria, apresentando contribuições no domínio do escalonamento da produção em duas envolventes principais: uma a um nível teórico, conceptual e outra ao nível prático da resolução de problemas. Ao nível conceptual contribui para uma ontologia de problemas de escalonamento da produção e conceitos relacionados, tais como ambiente de escalonamento de produção, tarefas, postos de trabalho, métodos de resolução, implementações e soluções, providenciando um enquadramento comum para a compreensão a partilha de conhecimento acerca destes conceitos. Ao nível da resolução dos problemas de escalonamento, faz-se um estudo alargado sobre a forma da sua resolução. Na concretização deste estudo desenvolve-se um sistema web de apoio ao escalonamento da produção (SWAEP), cuja arquitectura e funcionalidades são implementadas através de um demonstrador desenvolvido e implementado com base na tecnologia XML e na arquitectura de rede P2P, formando uma comunidade virtual de escalonamento da produção. Este demonstrador permite testar e avaliar a viabilidade das premissas e hipóteses subjacentes a este trabalho de investigação das quais é de realçar a pretensão de melhorar o processo de escalonamento de empresas industriais usando um

sistema web de escalonamento da produção baseados em conhecimento globalmente distribuído e acedido via Internet e intranets.

Esta filosofia de suporte ao escalonamento industrial pode considerar-se inovadora, porquanto difere das soluções existentes nas empresas ou actualmente acessíveis via Internet, por serem predominantemente centralizadas e pouco abrangentes.

Abstract

Production Scheduling is an important function strongly contributing to the competitiveness of industrial and service companies. It may be defined as the activity of allocating tasks to production resources, during a certain period of time. Solving a production scheduling problem involves several activities, starting with a clear description of it, with its identification within a class of problems. The next task is to find at least one suitable and efficient method, if it exists, for solving the problem, followed by running implementations of the method with the required set of input data and getting results in some adequate or desirable form. Often the people who have the problem to solve don't know about the methods or even the existing problem classification.

The research work here reported aims at facilitating the understanding of scheduling problems and improving the scheduling processes in companies. For this it presents contributions at two complementary levels. First, at the conceptual level it contributes for an ontology of scheduling problem related concepts, such as problem, scheduling environment, job, processor, method, implementation and solution, providing a common setting for knowledge sharing about these concepts. At the problem resolution level it comprises a web based scheduling system, with a distributed knowledge base, for sharing knowledge about scheduling and for aiding to solve scheduling problems. A demonstrator of this system is developed, implementing its architecture and functionalities, using XML technology and a P2P computer network forming a virtual community for production scheduling. The demonstrator is used for testing and evaluating the main idea behind this work namely that of sharing methods globally available for improving the scheduling process at companies by accessing these methods through the Internet. This scheduling strategy can be considered innovative having into account that most of the available scheduling systems are mostly centralized systems implemented at companies or accessed through the Internet and of limited scope.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas

AA	Amostragem aleatória
AWINQ	*Antecipated work in the next que*
BC	Base de conhecimento
BD	Base de dados
BL	Busca ou pesquisa em largura
BMP	Busca ou pesquisa do melhor primeiro
BP	Busca ou pesquisa em profundidade
BSP	*Batch sequencing problem*
BV	Busca ou pesquisa na vizinhanca
CO	Critério de optimizacão
COC	Critério de optimizacão complexo
CONR	Critério de optimizacão não regular
COR	Critério de optimizacão regular
EDD	*Earlist due date*
F/f1/PI	Sistema flexível de fase única e processadores paralelos idênticos
F/f1/PN	Sistema flexível de fase única e processadores paralelos não relacionados
F/f1/PU	Sistema flexível de fase única e processadores paralelos uniformes
F/fm/FP	Sistema flexível de fases múltiplas de linha pura
F/fm/JP	Sistema flexível de fases múltiplas de oficina pura
F/fm/OP	Sistema flexível de fases múltiplas de sistema aberto puro
f1/PI	Sistema de fase única e processadores paralelos idênticos
f1/PN	Sistema de fase única e processadores paralelos não relacionados
f1/PU	Sistema de fase única e processadores paralelos uniformes
FCFS	*First come at shop first served*
FM/f1/PI	Sistema flexível multi-processador de fase única e proc. paralelos idênticos
FM/f1/PN	Sistema flexível multi-processador de fase única e proc. paralelos não relacionados
FM/f1/PU	Sistema flexível multi-processador de fase única e proc. paralelos uniformes
FM/fm/FP	Sistema flexível multi-processador de fases múltiplas de linha pura
FM/fm/JP	Sistema flexível multi-processador de fases múltiplas de oficina pura
FM/fm/OP	Sistema flexível multi-processador de fases múltiplas de sistema aberto puro
fm/FP	Sistema de fases múltiplas de linha pura
fm/JP	Sistema de fases múltiplas de oficina pura
fm/OP	Sistema de fases múltiplas de sistema aberto puro

FO	Funcão objectivo
FOFO	*First off first on*
G	Sistema geral
G/f1	Sistema geral de fase única de processador único
G/f1/P	Sistema geral de fase única de processadores paralelos
G/fm	Sistema geral de fases múltiplas
G/fm/F	Sistema geral de fases múltiplas de linha
G/fm/J	Sistema geral de fases múltiplas de oficina
G/fm/O	Sistema geral de fases múltiplas de sistema aberto
GF	Sistema geral flexível
GF/f1	Sistema geral flexível de fase única de processador único
GF/f1/P	Sistema geral flexível de fase única de processadores paralelos
GF/fm	Sistema geral flexível de fases múltiplas
GF/fm/F	Sistema geral flexível de fases múltiplas de linha
GF/fm/J	Sistema geral flexível de fases múltiplas de oficina
GF/fm/O	Sistema geral flexível de fases múltiplas de sistema aberto
GFM	Sistema geral flexível multi-processador
GFM/f1	Sistema geral flexível multi-processador de fase única de processador único
GFM/f1/P	Sistema geral flexível multi-processador de fase única de processadores paralelos
GFM/fm	Sistema geral flexível multi-processador de fases múltiplas
GFM/fm/F	Sistema geral flexível multi-processador de fases múltiplas de linha
GFM/fm/J	Sistema geral flexível multi-processador de fases múltiplas de oficina
GFM/fm/O	Sistema geral flexível multi-processador de fases múltiplas de sistema aberto
GM	Sistema geral multi-processador
GM/f1/P	Sistema geral multi-processador de fase única de processadores paralelos
GM/fm	Sistema geral multiprocessador de fases múltiplas
GM/fm/F	Sistema geral multiprocessador de fases múltiplas de linha
GM/fm/J	Sistema geral multiprocessador de fases múltiplas de oficina
GM/fm/O	Sistema geral multiprocessador de fases múltiplas de sistema aberto
IA⌈D⌉	Inteligência artificial [distribuída]
LI	Limite inferior
LWKR	*Least work remaining*
M/f1	Sistema multi-processador de fase única de processador único
M/f1/PI	Sistema multi-processador de fase única de processadores paralelos idênticos
M/f1/PN	Sistema multi-processador de fase única de processadores paralelos não relacionados
M/f1/PU	Sistema multi-processador de fase única de processadores paralelos uniformes
M/fm/FP	Sistema multiprocessador de fases múltiplas de linha pura
M/fm/JP	Sistema multiprocessador de fases múltiplas de oficina pura
M/fm/OP	Sistema multiprocessador de fases múltiplas de sistema aberto puro
MAC	Módulo de aquisicão de conhecimento
MIU	Módulo de interface com o utilizador

MOPNR	*Most operations remaining*
MWKR	*Most work remaining*
PCV	Problema do caixeiro viajante
PD[D]	Programação dinâmica [para diante]
PL	Pesquisa local
PROLOG	*Programming in logic*
PSC	Programação sequencial clássica
PSE	Programação sequencial expandida
RD	Regras de despacho
RL	Ramificação e limite
RS	Regras de sequenciamento
S/ OPN	*Slack per operation*
SPT	*Shortest processing time*
TWORK	*Total work*
WIP	*Work in progress*

Lista de Símbolos

$\varnothing$	Símbolo de vazio, que indica uma característica por defeito, para cada
α	Classe de parâmetros α, que caracteriza o Ambiente de Produção
β	Classe de parâmetros β, que identifica as características dos
γ	Classe γ, que define critério de optimização
ΣN_{Ti}	Somatório (número total) de trabalhos atrasados.
(*)	Possibilidade de existência de qualquer instância do parâmetro em causa.
$\prec$, $\rightarrow$	Relação de precedência
ΣC_i	Somatório dos instantes de conclusão dos trabalhos.
$\Sigma w_i N_{Ti}$	Número total pesado dos trabalhos atrasados.
$\Sigma w_i C_i$	Somatório dos instantes de conclusão pesados dos trabalhos.
$\Sigma w_i T_i$	Somatório dos atrasos absolutos pesados dos trabalhos.
[]	Expressão ou termo facultativo
|	Barra de separação das classes de parâmetros ou barra de "ou".
agreg	Recursos agregados
aux_k	Recurso auxiliar k.
$avail_k$	Disponibilidade limitada dos processadores ou recursos auxiliares.
batch	Produção em lotes.
bm_k	Processador multi-item.
$buffer_k$	Armazéns locais de capacidade limitada.
chain	Precedências em forma de caminhos.
C_i	Instante de conclusão do trabalho i.
C_{max}	Instante máximo de conclusão dos trabalhos ou *Makespan*.
$comp_i$	Trabalho composto.
complete	Subclasse de problemas *NP-complete*.
Comp-pmtn	Interrupção complexa.
crt_k	Existência de recursos críticos.
d_i	Data para conclusão ou entrega do trabalho i.
$eleg_k$	Elegibilidade dos processadores ou recursos auxiliares.
fam	Produção de famílias de produtos.
free-pmtn	Interrupção livre dos trabalhos
F_{max}	Tempo de percurso máximo dos trabalhos.
F_{med} ΣF_i	Tempo médio de percurso dos trabalhos.
free-pmtn	Interrupção livre.

F_w	Tempo de percurso pesado dos trabalhos.
F_{wmed}	Tempo médio de percurso pesado dos trabalhos.
hard	Subclasse de problemas *NP-hard*.
L_{max}	Atraso máximo dos trabalhos.
L_{med}	Atraso médio dos trabalhos.
M	Conjunto dos processadores.
m	Quantidade de processadores.
M_k	Processador k.
mpm_k	Existência de processadores multi-função.
mpt_i	Trablho multi-processador i.
n	Quantidade de trabalhos.
n_i	Quantidade de operações do trabalho i.
no-wait	Inexistência de armazéns locais ou buffers.
NP	Classe de problemas *NP*.
O	Conjunto das operações.
O_{ij}	Operação j do trabalho i .
P	Classe de problemas *P*.
pmtn	Interrupção dos trabalhos ou das operações.
prec	Restrições tecnológicas ou de precedência.
R	Conjunto dos recursos auxiliares
r_i	Chegadas dinâmicas dos trabalhos ao sistema.
R_k	Recurso k.
s_{ik}	Preparação dos processadores ou recursos auxiliares para o trabalho i
s_k	Preparação externa dos processadores ou recursos auxiliares.
sp-graph	Precedências em forma de grafos.
T	Conjunto dos trabalhos.
T_i	Trabalho i.
t_{ij}	Tempo de processamento da operação j do trabalho i.
T_{max}	Atraso máximo positivo dos trabalhos.
T_{med} ΣT_i	Atraso absoluto médio dos trabalhos.
tree	Precedências em forma de árvore.
T_w	Atraso absoluto pesado dos trabalhos.
T_{wmed}	Atraso absoluto pesado médio dos trabalhos.
wait	Esperas ou tempos de inactividade nos processadores ou recursos
w_i	Peso, prioridade ou urgência do trabalho i.
Z ou γ	Critério de optimização.

1 Introdução

1.1 Enquadramento

De uma forma sintética podemos dizer que a actividade de escalonamento da produção numa organização procura fazer uso eficiente dos recursos de produção, com incidência predominante nos meios de produção e assegurar a rápida execução dos trabalhos por forma a fazer a sua entrega nos prazos acordados. Estes objectivos genéricos tendem a desdobrar-se numa variada gama de outros objectivos, avaliados por medidas de desempenho diversas, como por exemplo, o número médio de trabalhos ou encomendas atrasadas ou os lucros de produção por unidade de artigo vendido. Neste caso, o uso eficiente de matérias primas, dependente da sua utilização no tempo, é relevante ao processo de escalonamento e dependente deste. Esta dependência é claramente visível no OPT - *Optimized Production Technology* (Fox, 1983), e na filosofia Toyotista (Monden, 1983) ou da produção *Lean* (Womack, 1996), que tem subjacente a sincronização da produção com a procura de mercado em toda a cadeia de produção.

Apesar de ser geralmente reconhecida a elevada importância do processo de escalonamento na actividade produtiva de qualquer organização, a abordagem industrial ao escalonamento tende a ser simplista, resultando por isso frequentemente em soluções de qualidade modesta (Pinedo, 2002). Geralmente baseiam-se em abordagens empíricas orientadas à redução de custos preparatórios e à prioridade dos trabalhos. A razão de tal abordagem simplista resulta, aparentemente, por um lado, da falta de conhecimento da existência de métodos de qualidade que podem oferecer melhores soluções e, por outro, da dificuldade, conhecendo-os, de os implementar e utilizar na prática. Isto é verdade principalmente quando os métodos necessitam de ser implementados em, ou integrados com sistemas computacionais de apoio ao escalonamento da produção associados a sistemas ERP – *Enterprise Resources Planning* – para poderem ser utilizados pelas empresas. Esta dificuldade é portanto contornada recorrendo predominantemente a implementações de mecanismos simples baseados em regras de prioridade de execução aos trabalhos, como é exemplo a prioridade

baseada na urgência dos trabalhos em relação às suas datas de entrega acordadas ou em mecanismos heurísticos simples.

1.2 Motivação

Do acima exposto podemos concluir haver potencial para melhorar os processos de escalonamento de produção e desta forma a desempenho da empresa em várias dimensões, incluindo a da produtividade dos meios de produção e a do serviço ao cliente, talvez as duas dimensões mais influentes de sucesso e competitividade empresarial.

Dada a importância da função escalonamento e o actual cenário de inexistência de sistemas capazes de dar resposta adequada ao escalonamento da produção industrial, há necessidade de disponibilizar às empresas sistemas e ambientes de escalonamento apropriados às suas operações industriais.

Naturalmente que para proporcionar meios de escalonamento necessários e suficientes a uma empresa dos nossos dias poderia ser tentador, mas utópico, usar "super-sistemas" monolíticos e centralizados de escalonamento, dotados de um manancial de métodos capazes de lidar com as mais diversificadas exigências estáticas e dinâmicas de escalonamento. A isto estaria certamente associado um "super-custo", resultante quer da sofisticação dos sistema quer da demora provável para obtenção de boas soluções, que as empresas individualmente não poderiam suportar no mercado competitivo actual. Além disso tal abordagem traduzir-se-ia numa replicação ineficiente de recursos de gestão, à escala global. É, sim, recomendável usar sistemas focados e adaptáveis, capazes de resolver, nem mais nem menos do que as estritas e dinâmicas necessidades de escalonamento da cada empresa em particular.

Está-se a falar em sistemas flexíveis capazes de responder às necessidades de escalonamento específicas de cada empresa em cada instante, com métodos de resolução eficientes, adequados e ajustados a cada problema de escalonamento a resolver. Estes requisitos podem ser satisfeitos partilhando os métodos disponíveis numa flexível, dinâmica e expansível base de conhecimento globalmente distribuída, permitindo dar resposta, de forma também dinâmica, ao escalonamento necessário nos diversificados ambientes industriais e de mercado que podem ser encontrados. Este requisito genérico é incompatível com instrumentos de escalonamento monolíticos e dedicados a cada empresa e portanto exigem um corte com o passado, i.e. com lógicas de escalonamento tradicionais.

Uma forma, aparentemente eficiente e económica de escalonamento, que realiza o corte com o passado, é partilhar os mesmos métodos e recursos de escalonamento pondo-os ao serviço das necessidades de empresas diversificadas, de uma forma ubíqua, criando sistemas de escalonamento virtuais assentes numa base de métodos comum ou de agentes prestadores de serviços de escalonamento, de forma individual ou colaborativa. Isto constitui um novo paradigma designado de *paradigma ubíquo e colaborativo de escalonamento*. Num ambiente explorando este paradigma, vários prestadores de serviços de escalonamento contribuem para a realização de escalonamento, de qualidade, da actividade produtiva e logística de cada empresa. Podemos portanto conceber um sistema virtual de escalonamento, para cada empresa, criado a partir de uma base comum de agentes prestadores de serviços de escalonamento, associados a uma base globalmente distribuída de conhecimento de escalonamento da produção ou de métodos de escalonamento. Como características de tal sistema podemos identificar a sua virtualidade, a ubiquidade dos seus agentes, a sua adaptabilidade às necessidades de escalonamento de cada empresa/ rede de empresas, a expansibilidade e capacidade de actualização do seu conhecimento para melhor satisfazer as necessidades específicas de escalonamento em cada contexto industrial particular e em cada momento. Estes agentes prestadores de serviços podem basear-se em mecanismos ou algoritmos de natureza diversa, desde um métodos simples, implementado numa qualquer máquina acessível via Internet ou intranet, até sistemas de escalonamento mais complexos, emulando uma variedade de métodos capazes de resolver uma gama diversificada de problemas.

1.3 Objectivos

Tendo em conta o enquadramento e a motivação acima apresentadas, com este trabalho de investigação pretende-se abrir novas possibilidades para um melhor desempenho da actividade de escalonamento da produção nas empresas industriais, dando-se uma contribuição baseada em serviços web para a utilização de métodos de escalonamento da produção industrial distribuídos e implementados globalmente por fontes diversas. Desta forma é disponibilizado um espaço alargado de conhecimento de escalonamento potencialmente utilizável por qualquer empresa.

Pretende-se assim investigar a viabilidade de aplicar o paradigma de escalonamento ubíquo colaborativo, acima enunciado, para a concepção de sistemas web de apoio ao

escalonamento da produção (SWAEP) baseado em conhecimento distribuído de escalonamento.

Portanto, contrariamente às tendências verificadas recentemente de oferecer sistemas monolíticos poderosos centralizados, ainda que acessíveis via Internet, do que são exemplo sistemas tais como o *Lekin*, *flexible job shop scheduling system* (Pinedo, 1999) e o *Lisa*, *Library of scheduling scheduling algorithms* (http://lisa.math.unimagdeburg.de/), o *NEOS Server*, desenvolvido pelo *Optimization Technology Center*, da *Northwestern University* e do *Argonne National Laboratory* (http://www-neos.mcs.anl.gov/), o *ForthMP* desenvolvido pelo *Mitra's Group* da *Brunel University* (http://www.brunel.ac.uk/depts/ma/research/com), e o *IMS-NoE* (http://www.ims-noe.org/BENCHMARK/TBA.asp), a filosofia proposta é baseada na utilização de serviços distribuídos globalmente, centrada nas necessidades específicas de cada utilizador, que os utiliza de forma variável e de acordo com as suas necessidades em cada momento. Esta estratégia permite formar tantos SWAEP quantas as empresas ou utilizadores requerendo serviços de escalonamento. Dos utilizadores podem também fazer parte comunidades de ensino e investigação, ou simplesmente investigadores, académicos, industriais, que pretendam avaliar e estudar métodos especificados e existentes na web e compará-los com novos métodos que se vão desenvolvendo, ou simplesmente pretendam usar a rede para ensino ou formação no domínio do escalonamento.

Esta investigação tem por isso subjacente a definição de uma arquitectura genérica para os SWAEP capaz de ser instanciada por qualquer utilizador. Isto requer também um conjunto de instrumentos, interfaces e mecanismos capazes de concretizar a instanciação referida e proporcionar aos utilizadores o uso e acesso à base de conhecimento globalmente distribuída sobre métodos de escalonamento e ainda ao serviço de escalonamento associado à execução de métodos de resolução disponíveis através da Internet ou intranets.

A hipótese de que a filosofia de escalonamento proposta tem viabilidade, exige que o SWAEP seja no mínimo capaz de realizar ou oferecer as seguintes funções e funcionalidades:

- Especificar e representar problemas de escalonamento. Isto requer a utilização de um sistema de classificação e codificação de problemas.
- Especificar e univocamente representar métodos de escalonamento. Isto requer classificação e codificação de métodos baseada no sistema de classificação e codificação de problemas.

- Identificar métodos, a sua localização na web, capazes de darem solução a um problema a resolver. Esta função tem subjacente a necessidade de definir um esquema sistemático de associação de métodos a problemas de escalonamento e de poder obter, por exemplo, informação sobre o acesso para utilização do serviço de escalonamento associado a cada método.
- Especificar todas as variáveis necessárias e respectivos formatos para utilizar cada implementação informática de cada método, i.e. a sua *assinatura*. Isto é necessário para especificar cada instância de problema na forma que o método possa "entender" parar apresentar soluções.
- Ordenar a execução de cada método para obtenção de soluções. Isto é feito numa lógica de prestação de serviços web.
- Instanciar as variáveis da assinatura do método. Necessário para fornecer dados específicos de um problema a resolver utilizando o método.
- Especificar todas as variáveis ou medidas de resultados da utilização de um método na resolução de problemas. Isto é necessário para que o utilizador possa compreender e tratar as soluções dadas pelos métodos.
- Tratar resultados da solução de um problema, por exemplo apresentando-os num diagrama de Gantt ou tabela, ou inseri-los no sistema ERP do utilizador para posterior tratamento e utilização.

Estas funções serão analisadas e formas para possibilitar a sua realização serão estudadas através de procedimentos de especificação e representação tanto de problemas como de métodos, incluindo as suas assinaturas, e, ainda, de formas de representação de resultados dos métodos. A representação e especificação de classes de problemas e de métodos aponta para um sistema de classificação de problemas e outro, associado, de codificação, sendo por isso analisada a possibilidade de utilização de classes e codificações propostas por autores consagrados no domínio do escalonamento (Conway,1967 Blazewicz, 1996, Pinedo, 2002).

A hipótese de que a filosofia de escalonamento proposta tem viabilidade será testada e a verificada através da criação de uma instância de um SWAEP, com as funcionalidades acima descritas, que designamos de demonstrador. Desta forma pode-se instanciar e testar a validade de todos os mecanismos propostos e procedimentos desenvolvidos, quer para a especificação de problemas quer para pesquisa e selecção de métodos de resolução, quer para a identificação de agentes prestadores de serviços que tenham implementações dos

métodos de resolução seleccionados, quer, ainda, finalmente, para obtenção de soluções de instâncias dos problemas especificados pela prestação de serviços de escalonamento.

A especificação das assinaturas dos métodos de escalonamento que tenham implementações requerem a definição dos parâmetros que as caracterizam para que possam ser instanciadas com dados de problemas a resolver de forma a se utilizarem os métodos para obter soluções.

Face às características da linguagem XML- *extensible markup language* – e tecnologias associadas e, em particular à flexibilidade e interoperabilidade oferecida por esta tecnologia, será explorada a sua utilização nos procedimentos de especificação de problemas e de dados sobre problemas e métodos e, ainda, na definição de interfaces com o utilizador e de acesso aos serviços de escalonamento. Pretende-se, assim, concretizar a implementação dos conceitos fundamentais subjacentes à criação do SWAEP e desta forma uma abordagem computacional à resolução de problemas de escalonamento através da Internet. Complementarmente a criação da base distribuída de conhecimento de escalonamento, actualização dinâmica do repositório de conhecimento relativo a problemas e métodos, e o acesso a estes, serão baseados também na mesma tecnologia.

1.4 Estrutura da tese

Incluindo este capítulo, onde se faz uma introdução e definem objectivos, hipóteses e tarefas principais do trabalho de investigação, a tese está organizada, em 7 capítulos. O segundo, essencialmente baseado na revisão da literatura, está dividido em duas secções diferenciadas. Na primeira apresenta-se um breve enquadramento dos métodos com base em várias dimensões, incluindo técnicas subjacentes, princípios, abordagens, mecanismos e optimalidade das soluções que oferecem. Na segunda secção apresenta-se uma breve descrição de sistemas de escalonamento, com ênfase naqueles acessíveis pela Internet, fazendo-se a ponte para a razão de ser deste trabalho de investigação.

No capítulo 3 pretende-se identificar bem os parâmetros fundamentais e a estrutura de caracterização dos problemas de escalonamento. Começa-se por definir a função e ambientes de escalonamento. Caracterizam-se depois as entidades manipuladas e analisam-se os critérios de optimização geralmente equacionados. Evolui-se daqui para uma identificação e caracterização dos diferentes ambientes de produção, numa perspectiva alargada, estruturada e inovativa, apresentando-se também uma codificação estruturada para a identificação e representação de cada ambiente de produção. A natureza e

características das entidades de escalonamento e principalmente o ambiente de produção são críticos à identificação do tipo de problemas a resolver, equacionados pelos diferentes métodos de escalonamento, daí a importância trabalho que este capítulo apresenta.

A estrutura e caracterização desenvolvidas no capítulo 3 são a base para o desenvolvimento do trabalho descrito no capítulo 4, focado numa nomenclatura ou sistema de codificação capaz de poder representar e especificar todos os tipos de problemas de escalonamento a resolver. Começa-se para descrever e analisar as nomenclaturas e estruturas mais conhecidas, propostas por vários autores nas últimas décadas, avaliando a possibilidade de poderem ser utilizadas neste trabalho. A sua utilização é discutida originando como resultado uma proposta integradora considerada essencial para representar e codificar os problemas de escalonamento que poderão surgir na prática industrial. Esta nomenclatura inclui a codificação de ambientes de produção desenvolvida no capítulo 3.

No capítulo 5 faz-se uma revisão não exaustiva de métodos de escalonamento organizando-os e associando-os aos ambientes de produção identificados no capítulo 3 e codificando-os de acordo com a nomenclatura do capítulo 4. No essencial os métodos são sucintamente descritos, sendo também feita a sua associação a classes de problemas e identificada a referência bibliográfica onde é detalhadamente exposto ou é referido. Fica claro neste capítulo que a cada método pode ser associado um código identificador do problema que resolve. Este código na suas componentes estruturais identifica não só o ambiente de escalonamento mas também outras características importantes para a identificação do problema, relacionadas com as entidades envolvidas, tais como trabalhos, processadores e meios de produção auxiliares.

O capitulo 6 é o mais focado e relacionado como o objectivo central deste trabalho. Apresenta o trabalho principal da contribuição que se quer dar para a melhoria do processo de escalonamento nas empresas industriais baseado em serviços web e assente na utilização de métodos de escalonamento da produção industrial, distribuídos e implementados globalmente.

Esta contribuição possibilita a criação de sistemas web de apoio ao escalonamento da produção (SWAEP) baseados em agentes prestadores de serviços de escalonamento, através da Internet e intranets, distribuídos globalmente. No essencial descreve a arquitectura destes sistemas, assim como as suas funcionalidades e a forma como estas podem ser disponibilizadas. Estas funcionalidades foram referidas na definição dos objectivos do trabalho de investigação acima apresentada. Os instrumentos e mecanismos

necessários, desenvolvidos principalmente com base na tecnologia XML, possibilitam a instanciação de SWAEP nos diferentes utilizadores ou empresas e são a base para o desenvolvimento de um sistema de demonstração utilizado para testar e verificar a viabilidade da abordagem ao escalonamento que se propõe. Alguns testes são apresentados no fim do capítulo. Em particular, depois de implementadas todas as interfaces e mecanismos usa-se o demonstrador para resolver problemas teste de escalonamento, com base em métodos globalmente distribuídos, com análise e manipulação dos resultados oferecidos pelo serviço de escalonamento.

No capítulo 7 apresentam-se as conclusões deste trabalho de investigação. Este capítulo está dividido em duas partes fundamentais, uma primeira, onde se faz referência às contribuições deste trabalho para o apoio ao escalonamento da produção e uma segunda, onde se referem alguns dos desenvolvimentos futuros que se esperam vir a realizar.

A bibliografia referenciada nos diferentes capítulo aparece, por último, no capítulo 8.

2 Métodos e Sistemas de Escalonamento da Produção

2.1 Introdução

O objectivo deste trabalho foca-se na utilização de métodos de escalonamento da produção, que podem estar distribuídos, para a resolução de problemas de escalonamento. É, portanto, pertinente fazer uma abordagem sintética à diversidade de métodos que se podem encontrar e a sua caracterização. Desde logo, importantemente, podemos equacionar essa diversidade e caracterizar os métodos em função da sua aplicação e utilidade, i.e. da diversidade de problemas que podemos encontrar na prática e que, nos capítulos 3 e 4 são discutidos e classificados. Por outro lado, podemos olhar os métodos à luz de outros critérios de enquadramento ou classificação.

Assim, neste capítulo, faz-se uma síntese do enquadramento de métodos em função do tipo de programam que oferecem (secção 2.2) e de vários atributos, tais como, a natureza da solução obtida, em conformidade com o tipo de função de escalonamento equacionado (secção 2.3.1), a optimalidade da solução que proporcionam (secção 2.3.2), a natureza ou tipo de objectivo de escalonamento que equacionam (secção 2.3.3) e a técnica de optimização ou domínio de conhecimento em que é baseado (secções 2.3.4 e 2.4).

Dada o objectivo desta tese se centrar na proposta de um sistema de apoio ao escalonamento da produção, apresenta-se também neste capítulo um resumo de sistemas de escalonamento da produção existentes, tradicionais e de web (secção 2.5) e faz-se uma análise comparativa do sistema proposto com alguns destes sistemas afins existentes (secção 2.6). Por último, na secção 2.7 apresenta-se alguns comentários finais aos conteúdos expostos.

2.2 Tipos de programas de produção

Os métodos de escalonamento são úteis na medida em que podem ser usados para resolver problemas de escalonamento da produção que aparecem na prática industrial de produção de bens e de serviços.

Blazewicz (1996) classifica os problemas de escalonamento da produção como pertencentes a uma classe ampla de problemas combinatoriais designados de problemas de procura Π.

Definição 2.1 - Problema de procura Π: um problema de procura Π é um conjunto de pares (I, A), onde I designa a instância do problema, isto é, um conjunto finito de parâmetros (geralmente, números, conjuntos, funções, grafos) com valores específicos e A uma resposta ou solução para essa instância do problema (Blazewicz, 1996).

As definições seguintes, segundo Carsten Jordan (1996) são úteis para uma descrição mais precisa e uma melhor compreensão dos problemas em causa, cuja resolução conduz a uma solução, que será uma simples sequência de trabalhos a processar ou um programa mais elaborado e completo, com a especificação dos instantes de início e de fim de cada operação em cada processador.

Definição 2.2 - Sequência π: uma sequência $\pi = ((f[1], j[1]), (f[2], j[2]), \ldots, (f[k], j[k]), \ldots, (f[J], j[J]))$ consiste na atribuição de cada trabalho j ou (f, j), no caso de se tratar do $j^{ésimo}$ trabalho de uma família de trabalhos f, a uma posição k de uma lista ou vice versa.

Uma sequência representa apenas a ordem de processamento de um conjunto de trabalhos, enquanto que um programa contém, adicionalmente, os instantes de conclusão de cada trabalho, em cada um dos processadores.

Definição 2.3 - Programa σ: um programa $\sigma = (C(f[1], j[1]), C(f[2], j[2]), \ldots, C(f[k], j[k]), \ldots, C(f[J], j[J]))$ é um vector de instantes de conclusão de cada trabalho j, pertencente à família f.

Um programa pode ou não ser óptimo.

Definição 2.4 - Programa óptimo σ*: um programa óptimo σ* é um programa, tal que não existe nenhum outro programa com um melhor valor para a função objectivo do problema considerado.

Existem diferentes tipos de programas de produção, subjacentes aos seguintes conceitos (Baker, 1974):

Definição 2.5 - Programa ordenado: um programa ordenado é um programa em que os trabalhos ou lotes são executados pela mesma ordem em todos os processadores do sistema, pelos quais têm de passar, aquando da sua realização.

Sendo assim, para n trabalhos haverá, no máximo, n! programas ordenados possíveis.

Definição 2.6 - Programa não ordenado: um programa não ordenado, é aquele programa em que a ordem dos n lotes ou trabalhos nos m processadores pode variar. Neste caso haverá, no máximo, $(n!)^m$ programas não ordenados possíveis.

Definição 2.7 - Programa possível: um programa possível (para um determinado problema) é um programa que satisfaz as restrições do problema em causa; é, portanto, designado de possível "solução" para o problema e é aquele que é realizável na base dos requisitos tecnológicos e de capacidade produtiva existente.

Definição 2.8 - Programa activo: um programa activo, é aquele em que as operações começam logo que possível, sem atrasar qualquer outra operação ou violar as restrições de precedência, eventualmente existentes, entre operações (French, 1982).

Thomas Morton (1993) designa este tipo de programa de *Simple Dispatch Schedule* e define-o como sendo um programa em que os recursos nunca são mantidos inactivos, em antecipação à chegada de eventuais trabalhos urgentes ao sistema de produção.

2.3 Classificação dos métodos de escalonamento da produção

Os métodos de escalonamento da produção têm, por vezes, uma aplicação bastante alargada à resolução de grande variedade de problemas de escalonamento. Contudo, cada método é geralmente desenvolvido para resolver uma classe específica de problemas. Portanto, não só é pertinente como é necessário classificar os métodos e enquadrá-los nas classes de problemas que resolvem. Sendo esta abordagem de enquadramento e classificação dos métodos, à luz das classes de problemas, central ao trabalho desenvolvido nesta tese, focaremos esta vertente num capítulo dedicado. Aqui abordaremos a classificação e enquadramento dos métodos na base de outras dimensões classificativas.

A seguir referir-se-ão, mais detalhadamente, formas de classificação dos métodos, usando como parâmetro de classificação a natureza da solução obtida, a optimalidade da solução, o tipo de objectivo e o tipo de técnica subjacente ao método, respectivamente.

2.3.1 Classificação quanto à natureza da solução

A natureza da solução de um problema de escalonamento depende da natureza do método aplicado para a sua resolução, que incide num ou em vários processos ou funções de escalonamento da produção. Ou seja, cada método de escalonamento é vocacionado para dar resposta a uma ou mais funções do escalonamento. Assim, existem métodos

vocacionados para a resolução da componente afectação, em que se pretende determinar a melhor forma de atribuir trabalhos aos diferentes processadores disponíveis para a sua execução; casos em que se pretende abordar só, ou também, a questão do sequenciamento dos trabalhos, em que o interesse recai na ordenação dos trabalhos em cada um dos processadores disponíveis para a sua realização e ainda a calendarização, em que se pretendem determinar programas detalhados, com indicação dos instantes de início e de fim de cada trabalho, em cada um dos processadores. Combinações destas funções são também muito comuns, nomeadamente, afectação e sequenciamento.

2.3.2 Classificação quanto à optimalidade da solução

Métodos optimizantes

Os problemas de escalonamento pertencem a uma ampla classe de problemas combinatoriais. Um método de escalonamento é, genericamente, um método que constrói um programa de produção para um dado problema desta natureza (Blazewicz, 1996).

Geralmente está-se interessado num método optimizante mas, dada a complexidade subjacente a muitos problemas, utilizam-se muitas vezes métodos heurísticos, que não garantem a obtenção de uma solução óptima.

Os métodos de optimização são métodos que encontram sempre soluções óptimas para os referidos problemas, para um determinado critério de optimização considerado. Estes métodos são portanto designados optimizantes ou exactos (Morton, 1993). Trata-se de métodos em que, como a própria designação indica, se exploram as soluções possíveis para o problema, no sentido de encontrar a melhor solução, para o objectivo pretendido. Estes métodos têm normalmente subjacente uma complexidade exponencial, isto é, são métodos cujo tempo de resposta cresce exponencialmente, em função dos dados de entrada do problema (Blazewicz, 1996). Neste trabalho serão descritos, de um modo muito superficial, alguns métodos optimizantes, que permitem resolver muitos dos problemas combinatoriais, nomeadamente, métodos exactos de programação dinâmica e métodos exactos de "ramificação e limite", também designados de métodos de partição e avaliação ou métodos *branch and bound (B&B)*.

Métodos de aproximação e heurísticos

Dada a complexidade dos problemas de Escalonamento da Produção e a escassez de tempo de que geralmente se dispõe para a sua resolução, em muitos casos, não é possível

determinar uma solução óptima tornando-se imprescindível recorrer ao que geralmente se designam de métodos de aproximação.

Segundo French (1982) métodos de aproximação incluem aqueles em que se conhece quão próximas as suas soluções estão das óptimas e ainda uma variedade de outros métodos heurísticos que permitem obter boas soluções. Um método de aproximação "sub-óptimo" tenta apenas resolver, de uma forma aproximada, não tendo em vista a optimização de cada instância de um problema, não garantindo, portanto, que se encontre uma solução óptima (Morton, 1993). Embora as expressões "métodos heurísticos" e "métodos de aproximação" sejam muitas vezes utilizadas indistintamente na prática, eles reflectem realidades diferentes, na medida em que no primeiro caso não se tem, normalmente, uma ideia clara de quão afastadas as soluções encontradas se encontram da solução óptima, enquanto que no segundo caso se pode avaliar o grau de aproximação das soluções obtidas em relação à melhor das soluções possíveis, isto é, à óptima. Os métodos de aproximação são, portanto, normalmente, métodos com "precisão" analiticamente avaliada. Para alguns problemas combinatoriais pode ser provado que não existe esperança de se encontrar um método com uma precisão específica, uma vez que esta questão se torna tão difícil como a de encontrar um método de complexidade temporal polinomial para qualquer problema do tipo *NP*. Geralmente os métodos de aproximação são mais eficientes do que os optimizantes e mais fáceis de utilizar. Caso contrário seria preferível usar os métodos de optimização. Essa eficiência resulta normalmente da sua complexidade linear ou polinomial.

Um método de complexidade polinomial é um método cuja função de complexidade temporal é O(p(k)), onde p é um polinómio qualquer e k é o comprimento de entrada de uma instância de problema. Cada método cuja função complexidade temporal não pode ser limitada deste modo será designado método de tempo não polinomial, por exemplo de tempo exponencial.

Quando não é possível aplicar métodos optimizantes, de tempo polinomial, para a resolução dos problemas, devido ao facto de se tratar de problemas da classe *NP*, recorrer-se-á ao uso destes métodos de aproximação ou heurísticos que, portanto, não garantindo que se encontrem soluções óptimas para a maioria dos problemas, poderão conduzir a soluções consideradas boas e, muitas vezes, soluções próximas da óptima ou sub-óptimas (Ribeiro, 1999).

2.3.3 Classificação quanto ao tipo de objectivo

Embora a maior parte da investigação seja tradicionalmente focada em problemas de escalonamento com objectivo único, na prática, os problemas de escalonamento da produção mais frequentes são multi-objectivo. Recentemente, um número crescente de abordagens têm vindo a ser propostas e desenvolvidas para resolver problemas multi-objectivo.

Os métodos multi-objectivo são, portanto, particularmente interessantes, o que se torna particularmente importante para a resolução de problemas difíceis, nomeadamente, problemas que ocorrem em ambientes do tipo oficinas de fabrico, para os quais bastantes abordagens têm vindo a ser desenvolvidas.

Técnicas importantes, em que muitos de tais métodos se baseiam incluem programação dinâmica, *branch-and-bound* (ramificação e limite ou truncagem) e outras meta-heurísticas.

Exemplos típicos de meta-heurísticas, também conhecidas sob a designação de técnicas de pesquisa na vizinhança expandida (*extended neighbourhood search techniques*), são amplamente utilizadas e incluem algoritmos genéticos, *pesquisa tabu* e *simulated annealing* (Osman 1996, Arts 1997). Também merecem destaque outras abordagens bem sucedidas, frequentemente utilizadas na prática, baseadas em simulação, teoria de gargalos de estrangulamento (*bottleneck theory*), redes neuronais e redes de Petri, entre outras.

A seguir apresentar-se-á uma sintetização das principais técnicas subjacentes aos métodos de escalonamento, que geralmente são usadas na resolução de problemas que ocorrem na realidade industrial.

2.3.4 Classificação quanto à técnica de resolução

Uma forma muito típica de designar e classificar os métodos de escalonamento é quanto à técnica usada na pesquisa de soluções para o problema. A Figura 2.1 ilustra as principais classes de técnicas/ métodos de escalonamento, que se identificam na prática.

Os procedimentos matemáticos exactos são os métodos anteriormente definidos como métodos optimizantes, que têm por objectivo encontrar a solução óptima. Duas classes fundamentais destes métodos são os métodos exactos de programação dinâmica e os métodos exactos de "ramificação e limite ou *Branch and Bound*, *B&B*. Os métodos ou procedimentos heurísticos, que são métodos mais apropriados para a resolução de problemas de grandes dimensões, não garantem a obtenção de soluções óptimas, mas são

procedimentos mais expeditos e permitem, geralmente, obter soluções boas ou pelo menos aceitáveis e, em alguns casos, permitem chegar à solução óptima. Estes métodos têm sido cada vez mais explorados, como é o caso dos métodos de pesquisa na vizinhança ou pesquisa local, simples ou Expandida[1]. Destes fazem parte métodos de *simulated annealing*, métodos de pesquisa tabu[2] e métodos baseados em algoritmos genéticos. Outros métodos são de pesquisa em feixe (*beam search methods)* e há ainda métodos aproximados de programação dinâmica[3] e outros, nomeadamente, métodos aproximados de ramificação e limite e métodos baseados em técnicas *B&B* e merecem referência também os métodos baseados em gargalos de estrangulamento (*bottleneck methods)* (Morton, 1993).

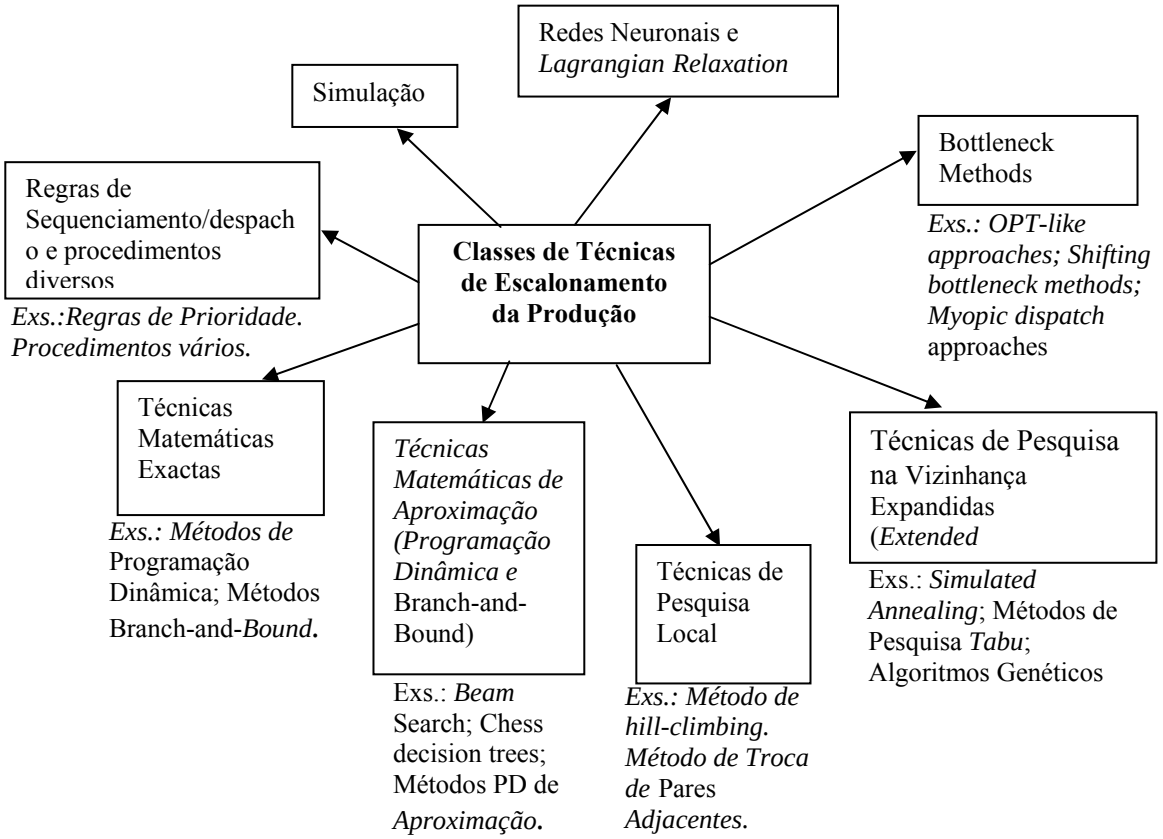

Figura 2.1 - Principais classes de técnicas/ métodos de escalonamento da produção.

2.4 Descrição de tipos de métodos e técnicas de pesquisa

Quando o espaço de soluções possíveis de um problema é reduzido será possível determinar e avaliar cada solução e pode ser apropriado seleccionar a mais vantajosa. Contudo, a explosão combinatória de soluções possíveis torna esta abordagem, de enumeração total das

[1] *Extended Neighborhood Search (ENS).*
[2] *Tabu Search (TS).*
[3] *Approximate Dynamic Programming (ADP).*

soluções, geralmente um processo impraticável. Poderá então recorrer-se sempre que possível a métodos de resolução rápida e eficiente, que são métodos heurísticos, por exemplo, métodos simples de pesquisa probabilística, como é o caso dos métodos de pesquisa aleatória (*random sampling*) ou *random walking*, ou através de heurísticos ou procedimentos que utilizam regras de sequenciamento ou de prioridade como, por exemplo, o método de Johnson (Conway, 1967) ou pela aplicação directa das regras de prioridade.

2.4.1 Métodos baseados em regras de sequenciamento

Uma regra de sequenciamento ou de despacho pode se entendida como um procedimento que permite ordenar os trabalhos, com base num determinado parâmetro para estabelecer prioridades nos trabalhos. Estas regras têm em vista atingir um determinado objectivo, expresso através de um determinado critério de optimização para o problema em causa. Estas regras podem ser simples ou combinadas, locais ou globais e estáticas ou dinâmicas (Baker, 1974). A seguir apresenta-se uma breve descrição e alguns exemplos típicos de cada uma destas classes de regras.

Regras locais *versus* globais

Nas regras locais a prioridade de afectação de um trabalho ou lote a um determinado processador é dependente, somente, da informação ou dados dos trabalhos na fila do processador em questão.

As regras locais são particularmente usadas nos modelos de simulação, referidos mais adiante.

Exemplos de regras locais são (Silva, 2006):

- *SPT – Shortest Processing Time*, dá prioridade ao trabalho à espera, cujo trabalho ou operação a executar a seguir tenha o menor tempo de processamento.
- *LWKR – Least WorK Remaining*, dá prioridade ao trabalho cujo tempo total de processamento das operações por executar seja o menor.
- *MWKR – Most WorK Remaining*, selecciona a operação associada ao trabalho que tem maior tempo de processamento a ser executado em operações a efectuar.
- *MOPNR – Most OPeratioNs Remaining*, selecciona o trabalho que tem o maior número de operações por executar.
- *RANDOM*, selecciona trabalhos aleatoriamente.

Nas regras de sequenciamento globais a prioridade é definida usando, além da informação local, a informação relativa a outros processadores.

Exemplos de regras globais (Silva, 2006):

- *AWINQ – Antecipated Work In the Next Que*, em que é dada prioridade à operação cuja operação que se lhe segue (do mesmo lote) será executada no processador onde o tempo total de processamento dos trabalhos à espera é mínimo. A lógica é não dar prioridade a um trabalho num processador que não pode ter andamento posterior, no processador seguinte, por estar sobrecarregada. Esta regra pressupõe que existe apenas um processador para realizar a operação seguinte do trabalho em causa.
- *FOFO – First Off First On*, em que é dada prioridade ao trabalho cuja operação possa ser concluída mais rapidamente. Se o trabalho não estiver na fila, o processador espera, parado, até que o trabalho chegue à fila para ser executada a operação.

Regras estáticas *versus* dinâmicas

Nas regras estáticas a prioridade de afectação relativa dos trabalhos no sistema não varia no tempo.

Exemplos paradigmáticos de regras estáticas são (Silva, 2006):

- *FCFS* (global) – *First Come/ arrival at the shop First Served*, em que é dada prioridade, ao trabalho que chega primeiro ao sistema.
- *TWORK – Total WORK*, em que a prioridade é dada ao trabalho cujo tempo total de processamento, de todas as operações, seja o menor.
- *EDD – Earlist Due Date*, que dá prioridade ao trabalho com menor data de entrega.

Nas regras dinâmicas a prioridade de afectação relativa dos trabalhos no sistema varia ao longo do tempo.

Exemplos típicos destas regras são (Silva, 2006):

- *S/OPN – Slack per OPeratioN*, em que a prioridade é dada ao trabalho que tiver o menor quociente do *Slack Time* pelo número de operações que faltam processar. *Slack Time* é o tempo que sobra para a data de entrega, depois de subtrair o tempo de processamento das operações ainda não processadas.
- *TSPT – Truncated SPT*, em que a prioridade é dada na base da regra *SPT* até que pelo menos um dos trabalhos da fila atinja uma espera determinada, W. W pode variar entre zero e um tempo estipulado.

Regras simples e complexas ou combinadas

Uma regra simples tem em vista a optimização de um determinado critério de optimização específico simples do sistema de produção em causa. Em algumas situações a razão da utilização de uma dada regra de sequenciamento desta natureza pode ser, simplesmente porque esta ajuda a aliviar o congestionamento dos trabalhos no sistema. Noutras circunstâncias poderia pretender-se implementar uma regra simples que permitisse satisfazer os prazos de entrega. Noutras, ainda, regras diferentes deveriam, possivelmente, ser escolhidas para outros objectivos diferentes.

As regras locais, globais, estáticas e dinâmicas anteriormente mencionadas são exemplos de regras simples.

Em alguns casos o uso de regras combinadas (uso combinado de duas ou mais regras simples) pode levar à melhoria dos resultados. Na prática, no entanto, o uso de regras combinadas é limitado. A razão principal é que, as regras combinadas requerem a determinação ou especificação do momento a partir do qual se deixa de usar uma regra simples e se passa a usar outra. Ora, a determinação de tal momento torna-se difícil avaliar perante cada situação particular.

Por exemplo, a regra *SPT* tende a deixar para trás os trabalhos de maior duração, por isso há necessidade de corrigir tal situação em casos reais. Desta forma, é frequente o uso da regra *TSPT – Truncated SPT –* em que se faz uso da regra *SPT* até que o tempo de espera, de pelo menos um trabalho na fila, atinja um dado valor. Se mais do que um trabalho atingir tal tempo de espera, então a regra *FCFS* é adoptada para esses trabalhos.

Algumas regras podem considerar-se simultaneamente estáticas e dinâmicas. Assim *SPT* e *LWKR* são estáticas relativamente a uma operação particular (por exemplo uma operação de 10 minutos de um dado trabalho terá sempre prioridade maior, relativamente a uma outra operação particular, de um outro trabalho qualquer, que demore mais de 10 minutos), mas são dinâmicas relativamente a um trabalho particular, na medida em que operações individuais, de um mesmo trabalho, adquirem prioridades relativas diferentes (por exemplo, um trabalho poderá ser primeiro que outro, num dado processador, situação que se poderá alterar em outros processadores, nas fases posteriores do processamento).

FCFS, pode também considerar-se simultaneamente uma regra estática e dinâmica, dependendo se é tomada em relação a cada processador do sistema, ou se é tomada em

relação ao sistema como um todo. Neste último caso a prioridade relativa dos trabalhos não se altera. No primeiro caso, será dependente da ordem de chegada ao processador.

Análise comparativa de regras

De acordo com determinados estudos efectuados, chega-se à conclusão que não se pode afirmar que determinada regra de sequenciamento "domine", ou seja, "é sempre melhor" do que as restantes. Uma regra que, contudo, tem revelado bastante sucesso é a regra *MWKR*, assim como algumas regras que derivam desta (Carvalho, 2000). Todavia, em determinadas situações, a regra *SPT* conduz a melhores resultados.

Em termos de tempo de percurso médio (F_{med}), como critério de optimização, também não se chegou à conclusão que uma regra dominava as outras, embora *SPT* e *LWKR* fossem normalmente mais eficientes do que as outras.

As experiências demonstram que a programação da produção em oficinas, baseadas em regras de sequenciamento é um procedimento praticável, para a obtenção de soluções sub-óptimas. Quando o critério de optimização é o instante máximo de percurso ou *makespan*, as regras com mais sucesso são as regras que exprimem a prioridade em relação ao trabalho que ainda é necessário levar a cabo em operações a realizar. Para os problemas do percurso médio (F_{med}), geralmente resultam melhor as regras que dão prioridade aos trabalhos com menor carga de trabalho, em termos de operações a realizar.

2.4.2 Métodos baseados em simulação

A Simulação é uma técnica de uso geral, que permite "imitar" ou recriar um determinado sistema real (por exemplo, um sistema de produção), através de um modelo, que consiste numa representação desse sistema real. Este modelo é concebido com a finalidade de reproduzir as características consideradas fundamentais ou mais relevantes desse sistema real e depende do estudo em causa (por exemplo, resolver um problema de escalonamento).

Esse estudo pode ser efectuado com o objectivo de projectar, compreender ou melhorar um determinado sistema, através da análise de um conjunto, mais ou menos vasto de variáveis, especificadas à partida, com a finalidade de extrair determinadas conclusões acerca do funcionamento ou comportamento que se espera obter para esse sistema real, a partir do comportamento do sistema modelado.

Os métodos baseados em simulação digital (MBSD) constituem uma classe de métodos que têm subjacente uma técnica utilizada num amplo conjunto de métodos e aplicações para

"imitar" o comportamento dos sistemas reais, através de uma determinada aplicação informática apropriada. De facto a Simulação é um termo bastante genérico, uma vez que a ideia subjacente a esta técnica se aplica em muitos campos, na indústria e nos serviços. A simulação é já uma técnica muito "popular" e tem-se revelado plena de potencialidades, com a evolução dos computadores e das aplicações informáticas, que estão sempre a melhorar.

A simulação, tal como outras técnicas, envolve então sistemas ou problemas, expressos através de modelos. A simulação por computador ou simulação digital é utilizada em métodos para estudar uma grande variedade de situações do mundo real, através de avaliação numérica ou quantitativa, usando aplicações projectadas para imitar o comportamento dos "trabalhos" e outras características relacionadas com o funcionamento do sistema, frequentemente, numa base temporal. Neste tipo de procedimentos há a necessidade de especificar um modelo para um sistema real, existente ou não, com o objectivo de conduzir experiências, de modo a obter uma melhor percepção e compreensão do comportamento desse sistema, sujeito a um determinado conjunto de condições ou restrições (Kelton, 1998).

Embora a simulação possa ser usada para estudar sistemas simples, o verdadeiro potencial desta técnica só se revela quando é usada para estudar sistemas relativamente complexos. Frequentemente, existem métodos alternativos aos MBSD e, contudo, opta-se, muitas vezes, por estes últimos, uma vez que os modelos de simulação permitem, muitas vezes, abordar problemas com níveis de complexidade tais que impossibilitam a sua resolução através de outras abordagens.

Os resultados de determinados estudos podem, normalmente, ser aplicados em situações similares. A extrapolação dos resultados para situações diferentes não é contudo aconselhável. Por isso é necessário bastante cuidado quando se pretende aplicar as conclusões de um dado estudo de Simulação a situações diferentes da que foi inicialmente analisada.

Estes métodos são muitas vezes aplicados a sistemas com relativa complexidade, nomeadamente em oficinas e em sistemas flexíveis de produção.

Um dos grandes benefícios do uso dos MBSD tem sido possibilitar o estudo do comportamento de uma grande variedade de regras de sequenciamento e identificar um

número reduzido de regras simples, mas eficazes, para cada sistema particular, com um objectivo específico.

2.4.3 Métodos de programação dinâmica

Métodos de PD exactos

Os alicerces da programação dinâmica (PD) foram estabelecidos por Bellman nos anos 50 (Morton, 1993). Neste tipo de métodos de resolução de problemas executa-se um procedimento de optimização, através de um processo "multi-estágio" e "recursivo", em que a decisão num estágio é requerida para a decisão no estágio seguinte, até se chegar ao estágio final, em que a solução do problema é obtida.

Quando a PD é aplicada a um problema de natureza combinatorial, então, no intuito de calcular o valor óptimo de uma critério de optimização, para qualquer problema ou subconjunto de tamanho k, terá, primeiro, de se conhecer o valor óptimo de cada subconjunto de tamanho k-1. Sendo assim, se o problema em causa é caracterizado por um conjunto de n elementos, o número de subconjuntos possíveis é 2^n. O que significa que os métodos de PD são de complexidade computacional exponencial. Contudo, para problemas difíceis, do tipo *NP-hard* é frequentemente possível construir métodos de PD pseudo-polinomiais que são de considerável relevância prática para instâncias de problemas de tamanhos razoáveis.

Métodos de PD heurísticos

Como refere Morton (Morton, 1993), nos métodos de PD heurísticos deve começar-se por efectuar programação dinâmica para diante[1] (PDD) com um procedimento exacto de pesquisa e resolve-se o problema, separadamente, através de um heurístico de elevada qualidade (por exemplo, troca de pares adjacentes (*adjacent pairwaise interchange*) ou *depth first beam search*. Um exemplo é o caso do método *ranking tolerance dynamic programming*, que recorre ao heurístico *apparent rankings*. Enquanto que os métodos de programação dinâmica para diante tem complexidade $O(2^n)$, o *ranking tolerance DP* tem complexidade $O(n(2b)^b)$, onde n é o número de trabalhos a serem sequenciados e b é o factor *ranking tolerance*, ou seja, um parâmetro do método *ranking tolerance dynamic programming*, que é um exemplo de um método aproximado de programação dinâmica (MAPD) (Morton, 1993).

[1] *Forward Dynamic Programming (FDP).*

2.4.4 Métodos de ramificação e limite

Os primeiros estudos de escalonamento aplicando métodos de ramificação e limite ou métodos de *Branch and Bound* (*B&B)* foram levados a cabo por Little et al (1963), citados em Morton (1993).

Estes métodos usam a técnica de ramificação e limite (RL). Esta técnica consiste, essencialmente, em começar por soluções parciais do problema e ir sucessivamente pesquisando a solução final, através dos caminhos possíveis de uma árvore invertida, de potenciais soluções do problema. Esta pesquisa faz-se, portanto, em passos sucessivos obtendo, em cada passo, soluções mais completas até que, no fim do procedimento de pesquisa se encontra uma solução completa e final, para o problema em questão.

"ramificação" significa subdividir um problema em dois ou mais problemas de complexidade mais reduzida e "limitar" é o processo de calcular um limite, superior ou inferior, para o valor da solução óptima, para cada subproblema gerado no processo de ramificação, de um valor directamente relacionado com o critério de optimização que se pretende considerar.

Métodos RL exactos

Nos MRL, supondo a existência de um conjunto finito S de soluções possíveis e um CO γ: $S \rightarrow R$, pretender-se encontrar $S^* \in S$, de tal modo que $\gamma(S^*)=\min/ \max_{(s \in S)}\{\gamma(S)\}$. Um MRL encontra S^* através de enumeração implícita de todos os $S \in S$ pela análise de subconjuntos cada vez mais pequenos de soluções de S (Brucker, 1995). Estes subconjuntos podem ser tratados como conjuntos de soluções de subproblemas correspondentes ao problema original.

O processo de ramificação pode ser convenientemente representado na forma de uma árvore de procura. No nível zero esta árvore consiste num único nó, representando o problema original, e em níveis subsequentes consiste em nós que representam sub-problemas específicos do problema no nível anterior. Nestes métodos são introduzidos e mantidos nós do problema, de cada nó do problema para cada um dos nós seguintes, também designados nós activos, correspondentes a sub-problemas que ainda não foram eliminados e cujos próprios sub-problemas ainda não foram gerados.

A escolha de um nó de um conjunto de nós gerados, que até então ainda não tinham sido eliminados nem deixados ramificar, é devida à estratégia de procura escolhida. Duas

estratégias de procura que são usadas mais frequentemente são a pesquisa em largura (PL) e a pesquisa em profundidade (PP). A PL implementa uma procura de fronteira ou em largura, onde é seleccionado um nó com um limite inferior[1] (LI) mínimo para ser examinado, enquanto que a PP implementa uma procura em profundidade primeiro, em que são examinados os nós descendentes de um nó parente ou progenitor próximo, tanto numa ordem arbitrária como no sentido dos LIs não-decrescentes. Assim, na estratégia de PL o processo de ramificação salta de um ramo da árvore para outro, enquanto que na estratégia PP procede primeiro directamente para o fim, ao longo de um determinado caminho, para encontrar uma solução inicial aproximada, isto é, uma solução experimental e depois refaz esse caminho para traz, de encontro ao primeiro nível, com nós activos. A estratégia de PP mantém relativamente poucos nós na lista em cada instante, comparativamente à estratégia de PL. Contudo, uma vantagem da PL é a qualidade das suas soluções intermédias que são, normalmente, mais próximas do óptimo do que as geradas pela PP, especialmente em estágios iniciais do processo de procura. Sumariando as anteriores considerações poderá dizer-se que, no intuito de implementar o esquema de um MRL, isto é, no sentido de construir um método de RL para um dado problema, terá de se decidir acerca de qual o:

- procedimento de ramificação e a sua estratégia de procura;
- procedimento de limitação, através de um critério de eliminação.

Uma vez tomadas as decisões anteriores poderá explorar-se a especificidade do problema e avaliar o compromisso entre a extensão do processo de ramificação e o "limite máximo de tempo" relacionado com o processamento dos LIs ou da solução experimental aproximada. O comportamento computacional dos métodos de RL mantém-se imprevisível e são necessárias muitas experiências computacionais para reconhecer a qualidade das suas soluções. A complexidade computacional temporal de um método de RL é exponencial, relativamente ao tamanho do problema, isto é, em relação aos dados de entrada, quando se procura uma solução óptima. Contudo, esta abordagem é frequentemente usada para encontrar soluções sub-óptimas, sendo possível obter complexidades temporais polinomiais fazendo parar o processo de ramificação, num determinado estágio do processo de pesquisa, após decorrido um certo intervalo de tempo. Esta estratégia é frequentemente aplicada nos métodos heurísticos.

[1] *Lower Bounds* (*LBs*).

Métodos de RL heurísticos

O processo de pesquisa através da técnica de RL pode ser de enumeração exaustiva, ou completa e, portanto, inadequado a problemas de grande dimensão. No sentido de se reduzir o espaço de pesquisa usam-se frequentemente métodos de RL heurísticos, que são normalmente baseados em (Morton, 1993):

- Funções geradoras: estas funções determinam a ordem de visita aos nós. São exemplos os procedimentos de PL e de PP.
- Funções avaliativas: estas determinam o valor, em cada nó, na pesquisa da solução óptima. Permitem assim implementar uma pesquisa em qualidade ou ordenada. Se o método heurístico utilizado permite obter soluções óptimas diz-se que a pesquisa é óptima. O processo de pesquisa é determinante na rapidez de obtenção da solução do problema. Em geral os heurísticos são mais eficazes na poda da árvore, isto é, na capacidade de excluir, ou 'podar', certos nós, que não encerram soluções potencialmente boas. Há métodos heurísticos que, no sentido de reduzir o espaço de soluções a observar, deliberadamente, não avaliam certos ramos da árvore, contentando-se com a obtenção de soluções normalmente sub-óptimas.

2.4.5 Métodos de pesquisa em feixe ou *beam search*

Os métodos de pesquisa em feixe ou *beam search* são métodos de enumeração parcial. Estes métodos permitem resolver problemas com base em árvores de decisão, no âmbito da inteligência artificial (IA) (tal como outros procedimentos heurísticos). É, portanto, um dos métodos desenvolvidos pela IA para árvores de decisão de enumeração parcial. Estes métodos também têm sido utilizados em muitos jogos de computadores. Todos eles utilizam uma mistura de estratégias de intensificação e de diversificação.

Trata-se de mais um tipo de procedimento de RL, em que, em vez de se desenvolver a árvore até se obter uma parte (ramo) da árvore que seja garantidamente sem efeito (sem utilidade), deverá prosseguir-se com ramos da árvore susceptíveis de serem "sem utilidade". É, portanto essencial ter uma boa medida para avaliar quão grande é essa susceptibilidade para ser um ramo sem efeito. Uma outra forma de proceder consiste em eliminar ramos, de modo a garantir poupança de esforços sem correr grandes riscos de perda de tempo na pesquisa efectuada (Morton, 1993).

O *beam search* está a um passo ou degrau da diversificação completa, subjacente à técnica de RL pura. A melhor forma de proceder consiste em focar a atenção na parte da árvore que

pareça mais vantajosa e os LIs dão uma boa indicação nesse sentido. Estes limites não dão o valor da solução esperada num determinado ramo mas apenas a melhor que daí se espera obter. Pode sempre optar-se por guardar mais ou menos subproblemas para análise, de tal modo que, alargando o espaço de análise se elevam os níveis de segurança e reduzindo esse espaço se obtém maior rapidez de resposta.

Sumariando as ideias básicas do *beam search* poderá dizer-se que se deve olhar para um procedimento exacto de procura, formar uma ideia acerca das áreas de procura desejáveis e indesejáveis e ignorar as áreas indesejáveis.

Beam search* / *best first search

É um exemplo de um método importante dentro da classe dos métodos heurísticos de RL, que é simples de aplicar e consiste, basicamente, na combinação de *best first B&B* e *beam search* e consiste em aplicar um método de ramificação e limite exacto, limitando o tamanho da tabela (árvore), por exemplo a 1000 nós. Neste método calculam-se ambos, o LI e uma solução experimental do *beam search* para o problema em causa. A solução experimental pode ser obtida, por exemplo, através de um procedimento de Pesquisa na Vizinhança[1] (PV).

Este procedimento "tenta" ser óptimo e deverá permitir encontrar soluções boas, com relativamente poucas expansões ou iterações. Em contrapartida, a validação das soluções obtidas por este procedimento torna-se um processo bastante moroso, comparativamente a outros procedimentos heurísticos mais simples (Morton, 1993).

2.4.6 Métodos de chess decision trees

Os métodos de *chess (playing) decision trees* têm semelhanças com ambos, métodos de ramificação e limite e métodos de *beam search.* Basicamente *chess (playing) programs* tentam aplicar a técnica de RL e eliminar ramos inferiores através de LIs. Contudo, existem, normalmente, muitos ramos a progredir em direcção ao fim na árvore, deste modo, o programa evoluirá, por exemplo, cerca de 8 a 10 minutos "para baixo" (pesquisa em profundidade), deixando um elevado número de subproblemas parcialmente desenvolvidos. O equipamento (*hardware*) e as aplicações informáticas (*software*) são obviamente de importância fulcral, para aumentar a velocidade da pesquisa (Morton, 1993).

[1] *Neighborhood Search.*

2.4.7 Métodos de pesquisa local e meta-heurísticas

Os métodos heurísticos de pesquisa local[1] ou métodos de pesquisa na vizinhança são métodos heurísticos que têm demonstrado grande aplicabilidade nos últimos anos. Os heurísticos de pesquisa local (PL) ou de pesquisa na vizinhança (PV) são muitas vezes designados meta-heurísticos ou estratégias com capacidades de engenharia do conhecimento e aprendizagem, reduzindo a incerteza, enquanto o conhecimento para os ajustes do problema é explorado e adquirido. No intuito de melhorar e acelerar o processo de pesquisa estes procedimentos tentam guiar, heuristicamente o processo de pesquisa para o melhor resultado (Morton, 1993).

Estes métodos funcionam em programas completos e tentam melhorá-los em cada passo do método e são designados de métodos de pesquisa local porque, normalmente, param quando encontram um óptimo local, e porque, em cada passo estudam a vizinhança local de uma dada solução completa. Para construir um método desta natureza terão de ser projectados 3 procedimentos, um procedimento que providencia um programa inicial possível, um procedimento que gera um programa na vizinhança, de um determinado modo, e uma função que providencia o critério de optimização de um determinado programa (Artiba, 1997).

Métodos de pesquisa local simples

Alguns dos aspectos mais importantes, com os quais se terá de lidar aquando da implementação de um procedimento de PL, consiste em tentar saber como obter uma solução inicial. Muitas vezes, encontrar uma solução inicial ou experimental não cria dificuldades, mas obviamente, a escolha desta solução poderá influenciar fortemente a qualidade do resultado final obtido. Deste modo, os métodos de PL deverão ser executados várias vezes, na mesma instância do problema, usando diferentes soluções iniciais, por exemplo, geradas aleatoriamente. A possibilidade de um determinado procedimento ser capaz de melhorar significativamente uma determinada solução depende, frequentemente, do tamanho ou espaço das vizinhanças. A escolha de vizinhanças para um problema é condicionada por uma análise de compromisso entre a qualidade desejada para a solução e a complexidade do método e é, geralmente, um problema resolvido através de experimentação. Outro aspecto crucial no projecto de um método de PL é a selecção de uma vizinhança ou vizinho que melhore o valor da Função Objectivo (FO).

[1] *Local Search Heuristics*.

Na realidade, a PL pode ser usada em problemas altamente complexos, para os quais os modelos analíticos envolveriam números astronómicos de variáveis e condições, ou acerca dos quais existe pouco conhecimento específico do problema disponível. Tudo o que é necessário com PL é uma definição razoável de vizinhanças, e um processo eficiente de as procurar. Quando estas condições estão satisfeitas, a PL pode ser implementada, para produzir, rapidamente, boas soluções, mesmo em amplas instâncias de problemas. Estas características da PL explicam a razão da sua aplicação numa vasta diversidade de situações.

A técnica do *hill-climbing* é uma técnica em que se encontram melhores soluções ao explorar soluções "próximas" de uma dada solução actual ou corrente e melhor até então encontrada. Esta técnica funciona bem num espaço de procura com relativamente "poucos" altos e baixos (*hills*), isto é, em espaços de procura pouco irregulares.

O *hill-climbing* iterado, de soluções aleatoriamente seleccionadas pode, frequentemente, atingir níveis de desempenho razoáveis, no processo de pesquisa, contudo, qualquer informação global avaliada durante a procura não será explorada. As técnicas de Amostragem Estatística são abordagens alternativas típicas que conduzem à acumulação e exploração de informação mais global. De um modo geral, poderá dizer-se que esta técnica opera dividindo iterativamente o espaço de pesquisa em regiões, a serem amostradas. As regiões que não sejam susceptíveis de conduzir a soluções aceitáveis são eliminadas, enquanto que as restantes são subdivididas para futuras amostragens. Se o número de sub-regiões úteis for pequeno este processo de procura pode ser bastante eficiente. Contudo, no caso de a quantidade de conhecimento no espaço de pesquisa, à partida, ser bastante reduzido, esta estratégia é frequentemente insatisfatória.

Infelizmente, a PL, na sua forma mais simples, o *hill-climbing* pára assim que encontra um óptimo local, isto é, uma solução x tal que $\gamma(x) \leq \gamma(y)$, para todos os y em N(x), em que N(x) é a vizinhança de x. Em geral, um tal óptimo local não é um óptimo global. Pior ainda, não existe normalmente garantia, de que o valor da FO, num óptimo local arbitrário, se aproxime do valor do óptimo global. Este inconveniente subjacente à PL pode ser ultrapassado, em alguns casos, através do recurso a múltiplos recomeços do processo de pesquisa. Mas dado que os problemas do tipo *NP* possuem, frequentemente, muitos óptimos locais, mesmo este procedimento poderá não ser suficientemente eficaz para obter soluções satisfatórias. Perante esta dificuldade, várias extensões à PL têm sido propostas,

que oferecem a possibilidade de obter óptimos locais através da aceitação de deteriorações ocasionais na FO.

A técnica de *hill-climbing* é uma estratégia de intensificação genérica, usada na Programação da Produção por Wilkerson e Irwin em (1971), citados em Morton (1993) e por muitos outros. Nesta técnica primeiro tenta-se encontrar uma solução inicial, por qualquer método (ex: método heurístico) uma vez que este método é melhor a refinar soluções do que na procura grosseira ou aproximada de uma solução inicial. De seguida deve-se tentar explorar todos os caminhos possíveis para alterar o programa suave e progressivamente, isto é, mantendo as soluções na vizinhança da solução corrente e avaliar cada programa resultante. Quando não houver mais nenhuma melhoria possível termina o processo de pesquisa de uma solução, caso contrário pega-se na melhor das melhorias, obtidas até então, e continua-se a pesquisa a partir desta, e assim sucessivamente. Esta constitui a base para muitos métodos modernos, tais como os baseados em pesquisa tabu e em *simulated annealing* (Varela, 1999).

É relativamente rápido e fácil programar e procurar um óptimo local, de modo que muitas vezes é surpreendentemente preciso, especialmente se for iniciado a partir de uma boa semente heurística. A sua principal desvantagem reside no facto de utilizar exclusivamente procedimentos de intensificação em vez de diversificação.

Esta técnica permite obter, geralmente de uma forma rápida, simples e flexível, boas soluções. Os elementos básicos, na aplicação desta abordagem ao escalonamento da produção, são os conceitos de vizinhança de uma sequência e um mecanismo para geração de vizinhanças. O mecanismo de geração de vizinhanças é um método que cria sistematicamente sequências relacionadas com uma sequência semente. Um exemplo de um mecanismo pode ser o da troca de pares adjacentes.

Em geral, dada uma semente e um mecanismo de geração, qualquer sequência que pode ser formada a partir da semente, numa única aplicação do mecanismo, é definida como pertencendo à vizinhança da semente.

Os métodos baseados na pesquisa de vizinhança são adequados para a obtenção rápida, de forma simples e flexível, de boas soluções.

Subjacente a tais métodos está:

- um mecanismo de geração de vizinhanças;
- a geração da vizinhança;

- um método para selecção de uma (nova) semente;
- a geração da sequência através de um procedimento de pesquisa local.

O método de troca de pares adjacentes (*pairwaise interchange method*) aparenta ser um método bastante rápido e preciso em muitos problemas, contudo este método geralmente apenas encontra óptimos locais e além disso existe pouco conhecimento acerca de como a mudança do problema afecta a solução. Mesmo assim, é muito útil, essencialmente se for utilizado em conjunto com bons métodos heurísticos iniciais, para extrair a solução inicial, como mencionado no método anterior de PL.

A seguir far-se-á uma abordagem muito resumida às extensões do conceito *hill-climbing* que são, *simulated annealing*, pesquisa tabu, *ejection chains* e algoritmos genéticos. Deverá mencionar-se que se trata de especificações particulares do que anteriormente se designou de conceitos de engenharia de conhecimento e de aprendizagem revistos em Morton (1993).

Métodos de pesquisa local expandida

Várias extensões ao conceito *hill-climbing* conduziram ao aparecimento de um conjunto de métodos que recorrem a técnicas designadas de pesquisa local expandida ou pesquisa na vizinhança expandida[1]. Trata-se de um conjunto de métodos que conservam as vantagens apontadas para os métodos do tipo PL/ PV simples, anteriormente referidos e que, além disso, são modificados no sentido de incorporar estratégias de diversificação. Alguns autores argumentam que estes métodos poderão ser considerados casos especiais de algoritmos genéticos. O método da amostragem aleatória (AA) ou *random sampling* e o *beam neighborhood search* são métodos de PL que diferem no processo de obtenção das sementes no processo de pesquisa.

O processo de AA é um dos tipos mais simples de pesquisa probabilística. Trata-se de um método de pesquisa aleatória que funciona simplesmente através da selecção aleatória de soluções, no espaço de pesquisa e devolve a melhor solução encontrada. O critério de paragem pode ser simplesmente um limite temporal, isto é, o processo de pesquisa termina quando se atinge esse limite temporal (Nurmela, 1993).

Outro procedimento parecido com a AA é o *random walking*. Neste processo, a única diferença deve-se ao facto de os movimentos efectuados, a partir de uma determinada solução actual, serem efectuados para uma das soluções vizinhas, numa determinada

[1] *Extended Neighborhood Search Techniques*.

vizinhança, em vez de se seleccionarem aleatoriamente as soluções, no espaço total de soluções disponíveis (Nurmela, 1993).

Métodos de pesquisa tabu

Tal como se referem Hertz et al, (1993), citados em Artiba (1997), as raízes da pesquisa tabu (PT) remontam aos anos 70. Esta técnica foi inicialmente apresentada, na sua forma actual, por Glover, em 1987, as ideias de base também foram esboçadas por Hansen, em 1986. A ideia original principal subjacente a este tipo de métodos é a de que é possível abandonar óptimos locais, através da aceitação de uma degradação no valor do critério de selecção de soluções, mas sem usar soluções previamente visitadas, que são armazenadas numa lista de soluções proibidas ou lista tabu (Artiba, 1997). Isto pode ser efectuado usando diferentes critérios, que assegurem que a perda incorrida no valor da função objectivo (FO), num tal processo de movimentação de "escape", não seja demasiado importante, ou de modo a que seja algures compensado.

A pesquisa tabu é uma meta-estratégia ou meta-heurística que se tem demonstrado ser um esquema eficaz e eficiente em problemas de optimização combinatorial, que combina uma estratégia de pesquisa do tipo *hill-climbing*, baseada num conjunto de movimentos elementares e um heurístico para evitar paragens em pontos sub-óptimos e a ocorrência de ciclos. Este objectivo é atingido através da utilização de uma lista finita de movimentos proibidos, ou movimentos tabu (a lista tabu), obtidos através do historial recente da pesquisa. A suposição de base a sublinhar neste procedimento é a de que os pontos sub-óptimos (onde o componente simples de *hill-climbing* pára) podem ser melhores pontos de partida, comparativamente a reinícios aleatórios. Deste modo, tenta-se evitar que o processo de pesquisa páre em óptimos (máximos ou mínimos) locais e também a possibilidade de entrar num processo cíclico de pesquisa (Battiti, 1996).

A pesquisa tabu é então uma meta-heurística baseada no uso de técnicas baseadas em proibição e em esquemas "inteligentes" como um complemento a métodos heurísticos de base, do tipo heurísticos de pesquisa local simples, com o objectivo de guiar o processo de pesquisa para fora de óptimos locais.

Existem pelo menos dois aspectos que caracterizam a generalidade dos métodos de PT, o facto de que esta técnica é usada para complementar o processo de PL e o facto de que as modificações à PL são obtidas através da proibição de movimentos seleccionados disponíveis a partir do ponto actual ou corrente. A PL é eficaz se a vizinhança for

apropriada à estrutura do problema, mas pára assim que seja encontrado o primeiro minimizador local, quando não existe nenhum movimento de melhoria disponível. A PT actua no sentido de continuar a pesquisa, para além do primeiro minimizador local, sem desperdiçar o trabalho previamente executado, como acontece se um novo processo de PL for iniciado, a partir de um novo ponto aleatório inicial. Usa então uma estratégia de diversificação para evitar que a trajectória de pesquisa permaneça confinada à proximidade de um determinado minimizador local (Battiti, 1996).

Na opinião de Battiti (1996), a principal vantagem, da PT, relativamente a heurísticos alternativos, baseados em PL, como os heurísticos de *simulated annealing* deve-se, precisamente, ao uso inteligente do historial da pesquisa, de modo a influenciar os seus passos futuros.

Os métodos de pesquisa tabu têm vindo a desenvolver-se, cada vez mais, no sentido de se tornarem métodos que recorrem a uma das estratégias gerais mais populares e com maior sucesso, na resolução de problemas de escalonamento. Sendo assim, analisando uma série de publicações referentes à pesquisa tabu, mais recentemente desenvolvidas, destacam-se, nomeadamente, as técnicas e*jection chains*, que serão referidas mais adiante (Petrovic, 2004).

Neste tipo de métodos cada iteração consiste em duas partes: o guia ou "processo tabu" e o processo de aplicação. O "processo tabu" actualiza a "lista tabu" e para isso necessita de uma lista actual (LA), que é uma lista ordenada de todos os movimentos (ou dos seus atributos) realizados ao longo da pesquisa, isto é, a LA representa o percurso das soluções encontradas e o processo de aplicação escolhe o melhor movimento, que não seja "tabu", e actualiza a LA.

A PT pode ser restringida ao subconjunto das soluções possíveis. Isto força ao que Glover chama uma "intensificação" da procura em permitir apenas "regiões" do conjunto possível. Uma ideia contrária também poderá ser usada para "diversificar" a procura. Nomeadamente, se todas as soluções encontradas numa fase inicial do procedimento de procura partilharem algumas características comuns, isto poderá indicar que outras regiões do espaço de pesquisa não tenham sido suficientemente exploradas. Identificar estas regiões não exploradas poderá ser útil para providenciar novas soluções iniciais para a pesquisa.

Métodos de arrefecimento simulado ou *simulated annealing*

Esta abordagem foi originalmente proposta, independentemente, por Kirkpatrick et al., em 1983 e por Cerny, em 1988, por analogia com o processo de fundição (*annealing*) de um sólido (Artiba, 1997).

O método *simulated annealing* é mais um tipo de método com aplicabilidade na resolução de problemas de optimização combinatorial, nomeadamente problemas de escalonamento da produção. O objectivo de um método desta natureza é encontrar a melhor solução de entre um número finito de soluções possíveis. A técnica de *simulated annealing* é uma técnica particularmente atractiva pois permite encontrar soluções próximas da óptima à custa de um esforço computacional pouco exagerado. É de notar que neste tipo de método, não é, geralmente, possível saber se a melhor solução encontrada, é o óptimo global. Esta característica restringe a utilização dum método deste tipo a casos em que um bom óptimo local é aceitável (Nurmela, 1993).

Os métodos baseados em *simulated annealing* são métodos que têm tido grande sucesso prático em muitos problemas de optimização combinatorial, do tipo *NP-complete*. Estes métodos podem ser entendidos como uma extensão dos métodos de pesquisa local probabilística simples. Se a nova solução proposta é igual ou melhor do que a solução actual então esta será aceite. Se a nova solução proposta é pior do que a solução actual será, mesmo assim, aceite com uma determinada probabilidade de aceitação.

Idealmente, quando a PL cai num óptimo local pobre ou fraco, esta técnica permite abandonar esse óptimo local.

A designação *simulated annealing* deriva de um processo físico. "*Annealing*" que é o processo físico através do qual um cristal é arrefecido, de um modo suficientemente cuidadoso, progressivo e lento. O estado energético do sólido, no final do processo de arrefecimento, estará no seu valor mínimo, ou muito próximo deste. Este processo pode ser entendido como um processo análogo à simulação do arrefecimento de um sólido.

A técnica de *simulated annealing* foi proposta como uma estrutura para a resolução de problemas de optimização combinatorial por Kirkpatrick, Gelatt e Vecchi e, independentemente, por Cerny (1985), citados em Morton (1993). Trata-se de uma técnica baseada num procedimento originalmente planeado por Metropolis et al. (1953), citados em Morton (1993), para simular o arrefecimento lento de sólidos, após estes terem sido aquecidos até ao seu ponto de fusão. Em procedimentos do tipo *SA*, as sequências das

soluções não tendem para um óptimo local, como acontece noutras técnicas de PL. Sendo assim, verifica-se que as soluções traçam um percurso ou trajecto aleatório, para cima e para baixo, através de um conjunto S, de soluções possíveis e este percurso tende a ser guiado numa direcção "favorável".

A técnica de *simulated annealing* tem sido aplicada a vários tipos de problemas de optimização combinatorial, com diversos níveis de sucesso. Regra geral, poderá dizer-se que esta técnica é um procedimento fiável para usar em situações em que o conhecimento é escasso ou se aparenta difícil de aplicar algoritmicamente. Mesmo para dar soluções a problemas complexos, esta técnica é relativamente fácil de implementar e normalmente executa um procedimento do tipo *hill-climbing* com múltiplos recomeços.

Esta técnica gera um caminho Markoviano, em que o sucessor de um ponto actual é escolhido estocasticamente, com uma probabilidade que depende apenas do ponto actual e não da história prévia da pesquisa. A propriedade não Markoviana é uma bênção misturada, ou seja, permite resultados heurísticos, que são bastante bons em muitos casos, mas torna a análise teórica do método difícil. Um método de *simulated annealing* é claramente desapropriado na resolução óptima de problemas de optimização combinatorial.

Métodos baseados em algoritmos genéticos ou evolucionários/ evolutivos

Uma outra extensão interessante à PL funciona com uma população de soluções possíveis (em vez de uma única) e tenta direccionar as propriedades que distinguem soluções boas de más. Estas propriedades são então usadas para construir uma nova população que se espera que contenha uma solução melhor do que a anterior. Esta técnica é conhecida sob a designação de algoritmo genético.

Existem muitas variantes de métodos de PL que têm vindo a ser propostos por vários autores, ao longo dos últimos anos. Segundo Charon e Hudry (1992), citados em Artiba (1997), os algoritmos genéticos são uns desses métodos, que surgem no intuito de ajudar a mover as soluções para fora dos óptimos locais, através do uso de perturbações ou, segundo Fleury (1993), citado em Artiba (1997), através de modificações na vizinhança, quando a probabilidade de estar num óptimo local atinge um determinado valor.

Holland (1975) e Goldberg (1989), citados em Artiba (1997), referem que, quando se usa uma população de soluções (em vez de uma única solução ou programa), em cada passo do processo de pesquisa, obtém-se um método de estratégia evolucionária, tais como os algoritmos genéticos. Este tipo de método funciona, basicamente, numa população, usando

três tipos de operadores, de "reprodução", de "cruzamento" e de "mutação", de um modo similar à vida das populações animais. Vaessens et al. (1994), citados em Artiba (1997) afirmam que esta família de métodos usa, de facto, uma hiper-técnica de vizinhança numa população de soluções em vez de uma única solução.

Tal como o nome sugere, os algoritmos genéticos são motivados pela teoria da evolução. Estes métodos remontam ao trabalho inicialmente descrito por vários autores, nomeadamente, Holland, em 1975 e Goldber, em 1989, citados em Morton (1993). Estes métodos foram projectados como estratégias de pesquisa geral e são métodos de optimização que funcionam sobre populações de soluções possíveis. Trabalhar com populações permite identificar e explorar propriedades comuns em soluções boas (isto é similar à ideia de "intensificação", mencionada na discussão da pesquisa tabu). As soluções são codificadas na forma de frases/ palavras ou *strings*, que consistem em elementos escolhidos de um alfabeto finito. De um modo grosseiro e aproximado, um algoritmo genético pretende produzir soluções próximas da óptima, permitindo a um conjunto de *strings* (alvos) representar soluções aleatórias, subjacentes a uma sequência de transformações unárias e binárias, geridas por um esquema de selecção com tendência para soluções de elevada qualidade. Deste modo, a qualidade do valor de um indivíduo numa população terá de ser definida numa *string*. Normalmente é o valor da FO ou alguma função derivada desta.

Os algoritmos genéticos não são, em geral, procedimentos bons no refinamento das soluções finais. Deste modo, pode usar-se uma abordagem híbrida, em que os algoritmos genéticos são usados no início do processo de pesquisa e outros procedimentos de PL são usados na parte final desse processo.

Por vezes a quantidade de cálculos necessários através do uso de métodos deste tipo, mesmo para obter soluções apenas próximas do óptimo global, são de tal ordem que tornam as pesquisas demasiado lentas. Contudo, estes métodos têm sido aplicados com sucesso em alguns casos mais simples (Nurmela, 1993).

Uma das dificuldades inerentes ao uso de algoritmos genéticos reside no facto de existirem imensos métodos alternativos desta natureza disponíveis, o que torna bastante difícil a avaliação de cada um deles, em termos de comparação do seu desempenho. Uma das principais vantagens deste tipo de método deve-se ao facto de serem bastante simples de implementar, mesmo em paralelo com outros procedimentos, por exemplo, métodos de aprendizagem (Nurmela, 1993).

As transformações nos indivíduos de uma população constituem os passos de reconstituição de algoritmos genéticos e são desempenhadas pelos três tipos de operadores anteriormente referidos. O efeito destes operadores é o de que as propriedades implicitamente boas são identificadas e combinadas numa nova população, que se espera que tenha a característica de que o valor do melhor indivíduo (representando a melhor solução na população) e o valor médio dos indivíduos sejam melhores do que em populações anteriores. O processo é então repetido até que seja atingido algum critério de paragem.

Através de uma população é gerada uma nova população temporária onde cada membro é uma cópia de um membro da população anterior. É então produzida uma cópia de um indivíduo, com probabilidade proporcional ao seu valor de adequabilidade ou *fitness*, sendo assim, é maior a probabilidade de melhores *stings* gerarem mais cópias. O efeito esperado desta operação é o de melhorar a qualidade da população como um todo. Contudo, não é criada nenhuma solução genuinamente nova e consequentemente nenhuma nova informação é acrescentada neste processo. A geração de tais *strings* novas é manipulada pelo operador de "cruzamento".

No sentido de aplicar este operador de "cruzamento", a população pode ser aleatoriamente dividida em pares. De seguida, para cada par, pode ser aplicado o operador de "cruzamento", com uma certa probabilidade, através da escolha aleatória de uma posição na *string* e alterando as "caudas" (definidas como sendo as s*ubstrings* na posição escolhida) das duas *strings* (isto é a versão mais simples de um "cruzamento").

O operador de "mutação", que efectua alterações aleatórias em elementos da *string*, desempenha, geralmente, apenas um papel secundário nos algoritmos genéticos. O operador de "mutação" serve para manter a diversidade na população.

Os algoritmos genéticos tradicionais ou clássicos, baseados numa representação (das soluções) em forma de *strings* binárias, é frequentemente desapropriada para problemas de optimização combinatorial, porque é muito difícil representar uma solução de tal modo que as *substrings* tenham uma interpretação plena de significado.

Os problemas de optimização combinatorial também caem dentro do âmbito dos algoritmos genéticos e inicialmente aproximavam-se do esquema a que Goldberg (1989), citado em Morton (1993) chama um algoritmo genético simples. Comparativamente a outros heurísticos poderá afirmar-se que, os algoritmos genéticos não são adequados ao refinamento de soluções, que estão muito próximas das soluções óptimas (Morton, 1993).

Enquanto que um método de ramificação e limite aprende a encontrar tais decisões conduzindo a uma solução óptima (relativamente ao espaço de todas as sequências de decisão) os algoritmos genéticos são capazes de guiar o processo de procura no sentido de aprender a encontrar as combinações de decisões mais promissoras, num intervalo de tempo aceitável. Deste modo, em vez de, implicitamente, enumerar todas as sequências de decisões será estabelecida uma árvore de pesquisa. Apenas um número polinomial de ramos será considerado onde as características das populações conduzem o processo de pesquisa para as regiões que mais provavelmente contêm soluções óptimas.

Métodos de profundidade variável/ *ejection chains*

Os métodos de profundidade variável, cuja terminologia foi popularizada por Papadimitriou e Steiglitz (1982), citados em Morton (1993), tiveram um papel importante nos procedimentos heurísticos para problemas de optimização. A origem de tais métodos remontam a protótipos de métodos em redes e teoria de grafos dos anos 50 e 60. Uma das classes destes métodos designados métodos de *ejection chains* mostrou-se bastante eficiente numa variedade de aplicações.

Os métodos do tipo *ejection chains* expandem as ideias exemplificadas por certos tipos de construções de "caminho mais curto" e "caminho alternativo". Os movimentos básicos para uma transição de uma solução para outra são movimentos compostos de uma sequência de passos emparelhados. O primeiro componente de cada par de passos numa abordagem de *ejection chains* introduz uma alteração que cria uma deslocação, isto é, uma "indução" para modificações futuras, enquanto que o segundo componente cria uma alteração projectada para recuperar o estado do sistema/ problema.

Os processos de *ejection chains*, não estão limitados à representação gráfica, em grafos de construções. Esta abordagem pode ser incluída numa implementação de pesquisa tabu completa ou numa implementação de algoritmos genéticos ou ainda numa implementação de *simulated annealing*. Um tal método também pode ser usado como um heurístico autónomo, que termina quando não é possível encontrar uma solução melhor, no final de qualquer um dos seus passos "construtivos".

Existem outras técnicas importantes, que merecem ser discutidas. Nomeadamente, métodos baseados na técnica de *lagrangian relaxation* e métodos baseados em redes neuronais. Estes últimos são mais recentes no domínio do escalonamento e por vezes ainda não

convenientemente testados para validar a sua eficácia e eficiência nestes domínios. A seguir apresenta-se uma descrição muito sumária destes dois tipos de métodos/ técnicas.

2.4.8 Métodos de relaxação

O método de relaxação de Lagrange ou *Lagrangian relaxation* consiste num caso particular de programação inteira simples, pela introdução de algumas restrições no problema às quais se associam determinadas penalizações, proporcionais às restrições ou violações incorridas. Este procedimento é combinado com um determinado procedimento de pesquisa (Morton, 1993). Este tipo de método é um "primo" dos heurísticos do tipo *bottleneck dynamics heuristics*. Porquanto muitas vezes poderoso, é de utilização complexa e ainda não sendo um método de uso muito generalizado, verifica-se uma tendência notória nesse sentido, nomeadamente aplicações em problemas de IO.

As vantagens associadas a este tipo de método, quando comparados com os métodos de programação inteira, advêm do facto de, normalmente, conduzirem a LIs "mais fortes", em problemas amplos e complexos de programação inteira, do que os métodos de programação inteira. Estes limites "mais fortes", conduzem a uma redução na procura, o que normalmente justifica a utilização destas técnicas, normalmente, também bastante mais morosas.

Estas técnicas também possuem outras desvantagens. É necessário dividir as restrições e eliminar apenas algumas delas. Este procedimento é um tanto delicado, na medida em que o problema resultante não seja demasiado fácil, por um lado, nem demasiado complexo, por outro, dado tratar-se de um problema de programação inteira, que terá de ser resolvido, com exactidão, muitas vezes, através da aplicação de métodos de relaxação de Lagrange. Existem outros aspectos críticos, nomeadamente, a melhor forma de procurar os "melhores" custos ou penalizações e a melhor forma de circundar ou isolar as soluções finais. Um método de relaxação de Lagrange é um método eficaz para abordar problemas inteiros do tipo *NP-hard*, que não são possíveis de resolver através de enumeração total ou através de métodos de ramificação e limite. A ideia geral subjacente à abordagem de relaxação de Lagrange consiste na decomposição de um problema amplo em problemas mais simples, Isto é possível através da simplificação das restrições ou constrangimentos do problema (por exemplo, restrições de capacidade), através do uso de multiplicadores de Lagrange (Kaskavelis, 1996).

2.4.9 Métodos baseados em redes neuronais

As redes neuronais têm a sua origem na IA. Tal como os algoritmos genéticos, anteriormente referidos, estes métodos tentam simular o evolucionário processo da selecção natural. As redes neuronais tentam emitar o processo de aprendizagem do cérebro humano, fundamentalmente a sua elevada conectividade (Russel, 1995).

Esta técnica tem atingido algum sucesso, em domínios muito diversos e há até quem defenda a ideia de se tratar de um "super-método" (Morton, 1993).

Um modelo de redes neuronais, também designado *connectionist network model*, consiste num conjunto de unidades computacionais, também designadas células, e um conjunto de conecções ou ligações unidireccionais juntando ou ligando as unidades (Gallant, 1993). Em determinados instantes uma unidade examina as suas entradas e faz a computação de um número com sinal, designado de activação, que é um resultado. A nova activação é então transferida ao longo das conecções, conduzindo a outras unidades. Cada conecção tem um número com sinal designado de "peso", que determina quão, uma activação que viaja ao longo desta conecção, influencia a célula receptora, na produção de uma activação similar ou diferente, de acordo com o sinal (+ ou -) desse "peso". O valor do "peso" determina a magnitude da influência da activação, de uma célula emissora até à célula receptora, sendo assim, quanto maior o valor (positivo ou negativo) de activação do emissor, mais significativo será o efeito na célula receptora. As conecções e os "pesos" são parâmetros importantes em qualquer modelo. Estes determinam o comportamento de um modelo e podem ser comparados a instruções simples num programa informático convencional.

Estes modelos foram motivados pela anatomia dos neurónios de seres vivos, com as células a corresponder aos neurónios, as activações a corresponder a taxas de activação neuronal e as conecções a corresponder a sinapses e os "pesos" a corresponder aos "esforços" sinópticos. Esta analogia não deverá, contudo, ser levada à letra. Estes modelos são, de longe, demasiadamente simples para servirem de modelos adequados ao processamento de informação que os organismos desempenham. Esta questão continua, contudo, em aberto, uma vez que existe um entendimento ainda um tanto limitado acerca do modo como o cérebro realmente funciona.

A verdade é que se trata apenas de um método não linear de previsão, operando com muitas das vantagens e inconvenientes dos métodos lineares de previsão, tais como, a regressão linear, entre outros. Estes métodos não são, geralmente, adequados à resolução de

problemas amplos de programação inteira, de um modo preciso, e não há motivo para se esperar que as redes neuronais o sejam (Morton, 1993).

Infelizmente, tal como acontece com os métodos de regressão linear, treinar uma rede neuronal, mesmo para um grande quantidades de dados disponíveis, não permite, muitas vezes, obter uma previsão perfeita. É verdade que quanto mais se treinar a rede neuronal, maior será a sua susceptibilidade para dar respostas a situações novas, ligeiramente diferentes das introduzidas e que não foram consideradas. Contudo, incorre-se num custo, que aumenta consideravelmente, à medida que se eleva o nível de exigência destas redes, no processo de reconhecimento e abordagem de situações novas. Além disso, coloca-se sempre a questão de quão novas, isto é, diferentes, estas poderão ser, de modo a serem convenientemente tratadas por esta técnica.

As redes neuronais estão a assumir cada vez mais importância e certamente continuarão a evoluir e a ser cada vez mais relevantes na resolução de problemas de escalonamento e de planeamento em geral.

2.4.10 Métodos de gargalo de estrangulamento ou bottleneck methods

Os métodos de gargalo de estrangulamento ou bottleneck methods dividem-se em decisões do tipo *Myopic*, para a sequenciação, que foram desenvolvidos nos anos 60, regras do tipo *OPT-like*, que apareceram nos anos 70 e inícios dos anos 80 e *bottleneck dynamics* que surgiram nos princípios dos anos 80 e 90. Versões dos 3 tipos de procedimentos podem ser utilizados para fazer estimativas de prioridades e tomar decisões de escalonamento das tarefas nos recursos, considerando um de cada vez (Morton, 1993).

Os *shifting bottleneck methods* constituem um tipo de método bastante utilizado na Programação de sistemas do tipo oficinas e foi inicialmente desenvolvido por Adams et al. em 1988 e posteriormente expandido por Balas e Vazacopoules. Outros trabalhos neste âmbito têm sido relatados por Ovacik e Uzsoy (1992) (Morton, 1993). Além disso estes métodos foram posteriormente melhorados por Dauzere Perez e Lasserre, em 1993, (Artiba, 1997).

Este método pode ser considerado de elevado custo computacional e uma versão de elevada precisão do *OPT*. Embora este método esteja essencialmente limitado ao objectivo *makespan* e a sistemas de produção de complexidade moderada tem, claramente, um grande potencial. Um caso particular deste método consistirá em considerar regras de decisão fixas, para todos os recursos excepto um que será o recurso crítico (*bottleneck resource*).

Deste modo, fixando uma determinada solução para um recurso crítico, aplica-se, repetitivamente, o procedimento subjacente ao método nos recursos restantes, para definir o "gargalo de estrangulamento" ou *bottleneck* entre os restantes e assim sucessivamente. À medida que o procedimento é repetido, a solução torna-se, eventualmente, possível (boa ou aceitável) para o problema global e o custo diminui até que, por fim, se encontra um óptimo local. Este método também se pode combinar com um determinado método de enumeração parcial, para obter melhores soluções.

Os métodos do tipo *shifting bottleneck* são métodos de melhoria iterativa que usam, repetitivamente, diferentes tipos de estratégia de simplificação do problema, de modo a torná-los problemas cada vez mais simples, por exemplo, problemas de processador único.

Basicamente, os métodos do tipo *bottleneck dynamics* permitem estimar custos associados a cada tarefa e aos recursos de um sistema de produção. Contrapondo os custos de atrasar as tarefas aos custos inerentes à utilização dos recursos, é possível calcular, por exemplo, qual a tarefa à qual está associada um maior rácio benefício/custo, de modo que essa seja programada para ser realizada de seguida, num determinado recurso, no sentido de minimizar a soma dos custos associados aos recursos e às decisões "parciais", em vez de globais, através de cálculos fáceis dos custos, permitindo obter soluções através de simulação. Os tipos de decisões que podem ser tomadas incluem diversos tipos de questões, nomeadamente estabelecer roteiros, questões relacionadas com os lançamento de encomendas em fabrico e programação detalhada ou escalonamento detalhado dos trabalhos, dimensionamento de lotes e agrupamento de trabalhos em famílias.

As *bottleneck dynamics* são particularmente indicadas para abordar casos em que diferentes processadores têm vantagens semelhantes para diferentes trabalhos. Quando se consideram questões do tipo dimensionamento de lotes e/ou problemas em que a preparação dos processadores depende da ordem de lançamento das ordens de produção, o primeiro passo consiste em converter tempos e/ou custos de preparação, de tal modo que os modelos de IO tradicionais e os heurísticos para dimensionamento de lotes possam ser aplicados.

Estes métodos também são de grande utilidade quando se está perante problemas que forçam a espera de processadores, em que se poderá chegar à conclusão que pode ser mais vantajoso esperar, isto é, manter os processadores inactivos, à espera de trabalhos urgentes ou prioritários, susceptíveis de estar prestes a chegar ao sistema.

Outros métodos de escalonamento da produção existem, que podiam também ter sido descritos, contudo, não é objectivo deste trabalho explorar detalhadamente este assunto, mas apenas apresentar uma revisão geral de alguns dos principais tipos de métodos existentes. Algumas referências bibliográficas onde podem ser encontrados mais desenvolvimentos a estes e a outros métodos incluem Bruker (1995), Blazewicz (1996), Jordan (1995), Pinedo (2002) e Vollmann (1997), entre muitos outros.

Para concluir o anteriormente exposto podemos afirmar que os métodos de escalonamento se baseiam em diferentes abordagens, que variam desde abordagens mais tradicionais, do âmbito da investigação operacional (IO), onde se têm desenvolvido métodos muito diversificados, nomeadamente métodos de programação matemática exactos ou de aproximação, incluindo métodos de programação dinâmica e de ramificação e limite (*branch and bound*), entre muitos outros.

Outras abordagens advêm de domínios da inteligência artificial (IA), como por exemplo, métodos baseados em redes neuronais ou em algoritmos genéticos ou evolucionários (evolutivos).

Existem ainda uma outra visão das abordagens, que variam deste abordagens mais simples, nomeadamente baseadas em bom senso e heurísticas diversas, até abordagens mais complexas, nomeadamente baseadas em simulação ou abordagens de IO/ AI mistas.

O objectivo do uso destas abordagens consiste em permitir um melhor suporte à tomada de decisão no escalonamento da produção, de modo a permitir aconselhar acerca de formas adequadas de coordenar as tarefas e os processadores, no sentido de realizar os trabalhos/ encomendas, num determinado intervalo de tempo.

Tais abordagens permitem então estabelecer um programa de produção será efectuado a diferentes níveis de detalhe e rigor, dependendo da função de escalonamento da produção a atingir e, consequentemente do tipo de decisão a tomar em cada instante, com base em determinados métodos usados para resolver os problemas em questão.

Na secção seguinte apresenta-se um resumo de alguns dos principais sistemas de escalonamento da produção, quer tradicionais quer de web, que utilizam abordagens e tipos de métodos que foram descritos nesta secção.

2.5 Sistemas de escalonamento da produção

Existem diferentes tipos de abordagens ao escalonamento da produção, resultando diversos tipos de sistemas, nomeadamente sistemas baseados em inteligência artificial (IA), ditos sistemas periciais, sistemas baseados em redes neuronais e algoritmos genéticos e ainda uma variedade de outros tipos de sistemas, nomeadamente sistemas de investigação operacional (IO), tais como, sistemas de programação matemática e diversos outros tipos de sistemas de suporte à tomada de decisão (SSD) no escalonamento da produção, incluindo sistemas baseados em abordagens mistas (sistemas IA/IO/SSD mistos) (Varela, 1999).

Estes sistemas têm aplicação em variados sectores industriais, podendo incluir vários métodos para a resolução de um conjunto alargado de problemas. Portanto, o seu grau de complexidade e abrangência podem ser variáveis e diversos.

O objectivo desta secção consiste em fazer uma breve revisão literária aos principais tipos de sistemas de apoio ao escalonamento, actualmente disponíveis, no intuito de contextualizar o trabalho desta tese.

Durante o início deste novo milénio vários sistemas têm emergido, baseados em abordagens novas e mais antigas, com especial incidência em novas tecnologias, onde merecem particular destaque as tecnologias de web e a Internet.

Alguns exemplos de sistemas de escalonamento da produção interessantes, que têm vindo a ser desenvolvidos ao longo das últimas décadas são, sumariamente, descritos a seguir.

2.5.1 Sistemas tradicionais

Por sistema tradicional pretende-se aqui considerar todos os sistemas que se baseiam em tecnologias convencionais existentes e que continuam a ser actualmente usadas, embora já em menor escala, para o desenvolvimento de sistemas de suporte à tomada de decisão no escalonamento.

Vários exemplos de sistemas de escalonamento da produção interessantes surgiram durante os anos 90:

LEKIN

O sistema LEKIN (Pinedo, 2002) é um exemplo para ambiente do tipo *job shop* flexível.

O desenvolvimento do sistema LEKIN decorreu de 1994 a 1996. Vários módulos foram transferidos para uma *software house*, que implementaram o sistema num site industrial. Em 1997 foi desenvolvida uma versão didáctica deste sistema.

O sistema LEKIN foi especificamente projectado para escalonamento de trabalhos num conjunto de ambientes diferentes, incluindo processadores paralelos, linhas e oficinas flexíveis. Também permite abordar a problemática das operações de montagem, isto é, uma unidade que recebeu processamento num processador e outra unidade que recebeu o processamento num segundo processador poderão ambas ir para um terceiro processador para serem montadas aí numa única unidade. O sistema suporta ainda tempos de preparação dependentes da sequência de processamento em todos os processadores e possui um calendário que permite ao utilizador determinar o número de movimentos associados a cada processador e introduzir períodos de manutenção e feriados. O sistema consiste num módulo de BD com um kit local e um módulo de interface com o utilizador. O módulo de BD gere os dados e funções na forma de um repositório para as heurísticas. Após uma sequência ter sido gerada a informação é enviada ao módulo de interface com o utilizador, que a permite visualizar de diferentes modos.

O sistema contém alguns métodos de escalonamento e heurísticas e foi projectado para permitir aos utilizadores testar as suas próprias heurísticas e comparar o seu desempenho com as heurísticas e métodos embebidos no sistema.

Além disso, é capaz de lidar com tempos de preparação dependentes da sequência de processamento dos trabalhos e todos os ambientes acima referidos. Este sistema permite manipular até 18 trabalhos, até 10 centros de trabalho ou estações, centros ou postos de trabalho e até 4 processadores em cada CT.

Os tipos de métodos incluem: regras de despacho, como EDD e WSPT, heurísticas do tipo *shifting bottleneck*, técnicas de pesquisa local, uma heurística para linhas flexíveis para o critério de optimização SumWjTj, isto é, minimização do atraso absoluto pesado dos trabalhos (*tardiness*).

O sistema permite construir manualmente programas de produção, através de uma interface gráfica, movendo os trabalhos nos diferentes processadores e visualizar as soluções através de diagramas de Gantt.

Além disso, o sistema também permite guardar e comparar diferentes soluções obtidas pela execução de diferentes métodos ou heurísticas, a fim de puderem ser comparadas.

LISA, Library of Scheduling Algorithms

LISA, Library of Scheduling Algorithms (Pinedo, 2002), disponibilizado no seu início em 1997, é um pacote informático para entrada, edição e resolução de problemas determinísticos de escalonamento, com particular incidência nos problemas de processador único. Este sistema pode ser usado para determinar a complexidade dos problemas e permite visualizar e manipular soluções obtidas, no intuito de apoiar no desenvolvimento de métodos adequados à resolução de determinados problemas.

LISA usa uma estrutura de dados própria para manipular problemas de escalonamento da produção, em termos das correspondentes entradas de dados e soluções. Este sistema disponibiliza um conjunto de métodos implementados, de ambos os tipos, exactos e heurísticos. Todos os métodos estão implementados externamente. O que significa que podem ser chamados e usados independentemente do núcleo do sistema (LISA core). É um sistema relativamente fácil de usar e permite adicionar métodos desenvolvidos pelo utilizador. A lista de métodos, cerca de 3 dezenas, disponibilizados incluem regras de prioridade e procedimentos diversos, incluindo métodos do tipo *branch-and-bound*, *shifting bottleneck*, *beam seach*, *neighborhood search*: *simulated annealing*, *threshold accepting*, *tabu search* e diversos outros procedimentos heurísticos e exactos.

Trata-se de um sistema gratuito, suportado em ambientes Linux, Unix e Windows. A maioria dos métodos disponibilizados foram implementados por estudantes. Este sistema é também usado para fins didácticos, no apoio ao escalonamento da produção e permite efectuar experiências por computador, para inserção de métodos, desta forma promovendo o estudo no domínio do escalonamento da produção. Permite, portanto, a transformação de problemas e a inserção de novos métodos.

Este sistema ainda não está disponível para ser usado de um modo directo através da Internet, embora possa ser adquirido e consultado através do seguinte endereço de web (http://lisa.math.unimagdeburg.de/).

Sistema de escalonamento desenvolvido por Lin e Lee

Lin e Lee, em 1997, desenvolveram um esquema para controlo e programação integrada de células flexíveis de manufactura, com base em redes de Petri. A metodologia proposta neste trabalho foi exemplificada através de uma célula flexível de manufactura preparada em Taiwan, nos laboratórios de investigação industrial de mecânica.

Sistema de escalonamento desenvolvido por Kazerooni et al

Kazerooni et al, em 1997, desenvolveram um sistema de suporte à decisão, baseado em lógica difusa, para a programação da produção em sistemas flexíveis de manufactura, usando simulação digital. Neste trabalho investigaram-se os problemas operacionais desses sistemas, através da combinações de regras de sequenciação, avaliadas através de um sistema de suporte à tomada de decisão integrado com lógica difusa.

Outros exemplos de sistemas de escalonamento da produção mais antigos e bem conhecidos existem, que emergiram em grande número, durante a década de 80, tais como: *YAMS* (Parunak, 1988), *SCORE* (Chiodini, 1996), *PATRIARCH* (Morton e Smunt, 1986), *MASCOT* (Erschler e Esquirol, 1986), *OPAL* (Bensana et al., 1986) e o *ISIS* (Fox, 1983), entre muitos outros.

YAMS - Yet Another Manufacturing System

YAMS - Yet Another Manufacturing System, de Parunak, em 1988, aborda o conceito de inteligência artificial distribuída e agentes inteligentes compostos. Trata-se de um sistema de planeamento e controlo da produção para ilustrar a utilidade e aplicabilidade da inteligência artificial distribuída neste domínio. Este sistema usa um modelo em rede para simular uma implantação típica da manufactura.

SCORE – Shop-floor COntingency Rescheduling Expert

SCORE – Shop-floor COntingency Rescheduling Expert é um sistema pericial desenvolvido por Chiodini, em 1986 e trata-se de um sistema de reprogramação em tempo real, que interactua com um sistema de planeamento de necessidades de materiais[1].

PATRIARCH

PATRIARCH é um sistema pericial desenvolvido por Morton e Smunt, em 1986, que permite resolver problemas de planeamento e programação de projectos em sistemas flexíveis de manufactura.

MASCOT

MASCOT é um sistema pericial desenvolvido por Erschler e Esquirol, em 1986 e é um sistema para programação de oficinas de fabrico, que usa uma análise do tipo análise baseada em restrições. O sistema visa manter os processadores constantemente disponíveis

[1] *Material Requirement Planning (MRP).*

e terminar os trabalhos dentro dos respectivos prazos. Este sistema foi concebido para resolver problemas de programação que envolvem apenas processadores e operações. Os instantes de início das operações e as restrições dos recurso são considerados aspectos importantes do problema. A abordagem de análise baseada em restrições é usada para gerar relações de precedência em recursos conflituosos.

OPAL

OPAL, um sistema desenvolvido por Bensana et al., em 1986, é um sistema pericial para programação da produção similar ao anterior e foi apresentado em 1986. Este sistema de programação da produção em oficinas de fabrico utiliza uma representação de conhecimento em forma de regras de produção.

ISIS

ISIS, desenvolvido por Fox, em 1983, na Universidade de Carnegie-Mellon, é também um sistema pericial bem conhecido para programação em oficinas de fabrico. Este sistema é baseado em linguagem de representação por esquemas *(schema representation language - SRL*), uma linguagem de representação de conhecimento baseada em *frames*. O esquema (*frame*) providenciado em SRL consiste no nome da *frame* e um conjunto de campos. No processo de geração do programa várias condições são usadas para direccionar o procedimento de pesquisa. Estas condições consideram os parâmetros prazos, recursos e custos.

2.5.2 Sistemas de web

Os sistemas de web, na Internet são sistemas que surgiram na sequência dos anteriores, sistemas tradicionais ou mais convencionais e são sistemas que, além de outras vantagens, podem ser facilmente usados e partilhados em rede.

A seguir apresenta-se alguns dos sistemas desta natureza que foram desenvolvidos nas duas últimas décadas.

NEOS Server

O NEOS Server, resultado colaborativo do Optimization Technology Center, da Northwestern University e do Argonne National Laboratory, é um sistema de web que pode ser usado para a resolução de problemas de optimização, incluindo problemas de escalonamento da produção, com particular destaque para um tipo particular de problema

que é o problema do caixeiro viajante (*Travelling Salesman Problem* - TST). Este servidor disponibiliza perto de 50 programas "solvers" através de uma vasta variedade de interfaces em rede (http://www-neos.mcs.anl.gov/).

BBN's Vishnu

Um outro sistema que merece destaque é o BBN's Vishnu. Trata-se de um sistema de escalonamento, baseado na web (http://vishnu.bbn.com). O sistema Vishnu é um mecanismo poderoso de escalonamento para diversos cenários práticos reais. Possui um módulo facilmente adaptável para realizar diversas formas de escalonamento, desde optimizar operações e serviços, planeamento estratégico logístico, incluindo escalonamento de processos. O Vishnu inclui interfaces de utilizador para visualização e edição de programas e de dados.

Este sistema foi desenvolvido como parte de uma investigação levada a cabo em algoritmos genéticos e técnicas de programação evolutiva ou evolucionária. É um sistema que tem subjacente uma forma de representação dos problemas em ambiente que permite aos utilizadores especificarem o problema que pretendem resolver, sendo portanto reconfigurável.

O sistema automático do Vishnu ("schedulers") utiliza pesquisa genética para encontrar programas optimizados. É um sistema fácil de integrar dado que as entradas e saídas e as especificações dos problemas são todas especificadas através de *xml schema*.

Além disso, é um sistema colaborativo, na medida em que os "schedulers" podem colaborar, através de uma plataforma de agentes ou funcionar de um modo autónomo e individual.

O sistema também possui um modo de funcionamento na web, que permite interacções em rede com os "schedulers" e os dados.

Este sistema também suporta o desenvolvimento rápido de protótipos, com um GUI[1] autónomo, que mostra programas, dados e vistas personalizadas sobre os dados.

O Vishnu permite efectuar escalonamento optimizado para um vasto conjunto de aplicações de escalonamento integrando as seguintes características: é um sistema de escalonamento com uma ampla aplicabilidade e significativamente reduzidos custos de desenvolvimento e implementação:

[1] Graphical User Interface.

tem independência do domínio, isto é, a maioria dos programas "schedulers" resolvem uma limitada classe de problemas de escalonamento num único domínio. Em contrapartida o enquadramento de representação de problemas do Vishnu permite aos utilizadores especificarem o problema a ser resolvido e adaptar-se ao domínio em causa. Por exemplo, permite escalonar serviços de táxi, da mesma forma que permite escalonar horários de salas de aula ou serviços de visita. A flexibilidade deste sistema para se adaptar a requisitos específicos dependentes do contexto confere-lhe um certo potencial, sendo um dos seus principais atractivos.

Optimização: Um programa ou "scheduler" usa pesquisa genética para definir programas optimizados. A maioria dos problemas de escalonamento incluem ambos, restrições fortes ou absolutas, que têm de ser forçosamente cumpridas, e restrições mais leves que podem ser violadas sob um determinado custo ou penalização. À medida que os problemas de escalonamento se tornam mais difíceis a quantidade de compromissos que têm de ser considerados aumentam. O escalonamento optimizado avalia os custos desses compromissos para identificar uma possível solução óptima.

Modo de servidor de web: Neste modo todas as interacções com o sistema são efectuadas através de um servidor de web. Os utilizadores podem usar um browser de web normalizado para definir problemas de escalonamento, ver programas e controlar quando o programa automático deverá correr. As páginas de web, dinamicamente criadas, possuem as seguintes características: Diagramas de Gantt; vista e acesso integral a todos os dados; edição de dados e de programas; hiper-ligações e navegação fácil; dados externos de uma BD partilhada em rede, através do uso de uma interface XML própria, definida para o efeito.

Integração: entradas, saídas e especificações de problemas são especificadas por esquemas XML (XML schemas) no programa Vishnu reconfigurável.

Colaboração: Múltiplos programas Vishnu podem colaborar quer na forma de processos autónomos em Java ou num enquadramento de agentes.

Desenvolvimento rápido de protótipos: o Vishnu Scheduler GUI mostra e permite editar programas, dados e regras de escalonamento e personalizar a visualização de dados. Tem associado um programa interno, que permite o desenvolvimento rápido de protótipos enquanto regras de escalonamento estão a ser desenvolvidas.

FortMP

O FortMP (http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGide/Blurbs/fortlp.html) é um sistema de programação matemática, do Mitra's Group, da Universidade Brunel University, um grupo de investigação de programação matemática, dirigido pelo Prof. G. Mitra (http://www.brunel.ac.uk/depts/ma/research/com) e está acessível, entre muitos outros, através do anteriormente referido Neos Server.

Este sistema foi projectado para resolver problemas de programação linear, programação quadrática, programação inteira entre outros tipos específicos de problemas de programação matemática (como a mista: inteira e linear), através de um conjunto de algoritmos disponíveis, que incluem o SIMPLEX, PRIMAL e DUAL, entre outros, nomeadamente do tipo branch and bound. É um sistema projectado para correr na maioria dos processadores. Este pacote está a ser disponibilizado na forma de código-fonte. É totalmente suportado e mantido, com actualizações a serem produzidas regularmente. O FortMP aceita dados de problemas em formato MPSX mas está desenvolvido de um modo modular, por forma a permitir aos utilizadores usarem os seus próprios métodos/algoritmos para entrada de dados de problemas de optimização.

e-OCEA

O e-OCEA (http://www.ocea.li.univ-tours.fr/eocea/index.jsp) é um portal para escalonamento que pretende ajudar na identificação de problemas de escalonamento da produção, ajudar no desenvolvimento de novos métodos e conduzir fóruns de discussão através da Internet.

É um portal web para investigação no domínio do escalonamento, incluindo métodos, referências e dados de problemas. Trata-se de um projecto levado a cabo por uma equipa de escalonamento do Computer Science Laboratory, University of Tours. É um projecto suportado pelo grupo de Investigação Operacional do National Centre of Research CNRS, em França.

Este portal está a ser desenvolvido no contexto de um projecto em curso, que se iniciou em meados dos anos 90. Alguns componentes do sistema estão já completamente operacionais outros estão ainda em fase de desenvolvimento. A gestão das referências, conjuntos de dados e métodos funciona, embora persistam alguns problemas por resolver. A componente de referências evolui à medida que utilizadores do sistema adicionem o seu próprio

material. Até ao momento foram introduzidas cerca de 4000 referências bibliográficas no sistema.

O componente composto pelos módulos está em progresso e a maioria dos oferecidos são apenas protótipos.

Este portal de escalonamento contém uma BD de métodos, referências e conjuntos de dados trazidos pela comunidade de escalonamento. No futuro serão adicionadas ferramentas para ajudar a identificar problemas de escalonamento, ajudar no desenvolvimento de novos algoritmos e conduzir fóruns de discussão através da Internet.

O objectivo deste portal é o de gratuitamente providenciar uma plataforma para escalonamento.

LekiNET

O LekiNET, um protótipo de um ambiente de escalonamento na Internet, por Benjamin P. C. et al. (Yen et al., 2004), que migrou do LEKIN/ spl reg, um sistema para escalonamento em job shop flexível, que tem algumas similaridades com o sistema aqui proposto, focando mais em escolhas vantajosas do ponto de vista de custos económicos, de agentes de escalonamento para a resolução de problemas.

Trata-se de um ambiente de escalonamento formado por um grupo de agentes de escalonamento na Internet que partilha recursos computacionais para resolver problemas de escalonamento de um modo distribuído e colaborativo. Os autores propõem um esquema migratório para transformar sistemas autónomos existentes em agentes de escalonamento na Internet que possam comunicar entre si e resolver problemas para além das suas capacidades individuais. De modo a coordenar a colaboração de recursos computacionais entre agentes, os autores introduziram um mecanismo de controlo baseado no mercado no qual agentes interessados iniciam ou participam em audições de venda e compra de problemas de escalonamento. As afectações dos recursos computacionais é conseguida através destas audições ou auditorias.

Através de experiências levadas a cabo com este sistema os autores concluem que o ambiente de escalonamento baseado em agentes e estratégias de negócio de mercado é viável e vantajoso para pesquisas de escalonamento futuras e futuros desenvolvimentos.

Riot (Remote Interactive Optimization Testbed)

O sistema Riot (Remote Interactive Optimization Testbed), que está disponível através do endereço: http://riot.ieor.berkeley.edu/riot/Applications/Scheduling/index.html, é decorrente de um projecto conduzido por um grupo de investigadores orientado pelo Professor Dorit S. Hochbaum e é suportado pela National Science Foundation, através do seu apoio à investigação.

Através deste site de web os autores do sistema oferecem ao utilizador a possibilidade de experimentar diferentes combinações de objectivos, ambientes de produção (tipos de sistemas de produção) e características de problemas, através da colocação dos seus próprios problemas de escalonamento a resolver, testando o desempenho de vários métodos, quer óptimos quer heurísticos. Em geral os métodos são procedimentos relativamente simples ou regras de despacho/sequenciação, mas permitem diferentes combinações de método e objectivo a testar, desde que não haja conflitos entre as entradas de um método e o objectivo seleccionado. O sistema tem um índole essencialmente didáctico e também permite avaliar a complexidade dos problemas definidos no sistema, com base no nomenclatura de classificação de problemas proposto por Brucker e na sua BD de referências de métodos de escalonamento da produção.

2.6 Análise comparativa de sistemas

Muitos artigos têm vindo a ser publicados sobre sistemas aplicados ao escalonamento da produção. Estes sistemas incluem sistemas comerciais de apoio ao escalonamento e também sistemas de índole mais académica e pedagógica, não visando, geralmente, a aplicação prática real.

Recentemente tem-se verificado uma tendência para um aumento dos sistemas de escalonamento da produção acessíveis através da Internet.

Os sistemas de web actualmente disponibilizados em rede envolvem, geralmente, vários métodos, para a resolução de uma gama de problemas, usando técnicas específicas, tais como programação matemática. Além disso, tais sistemas não estão, em geral, projectados para facilmente incorporar novo conhecimento de escalonamento e novas implementações de métodos, nomeadamente através de utilizadores do sistema.

O NEOS Server, anteriormente referido, é um sistema que, de acordo com os autores, embora tenha evoluído com a web e a Internet, está, de certa forma, limitado por decisões de projecto tomadas anteriormente (http://www-neos.mcs.anl.gov/).

Este servidor de métodos disponibiliza perto de 50 programas através de uma variedade de interfaces de rede. Os programas estão implementados em diferentes linguagens de programação e requerem interfaces específicas para que possam ser executados e dar resposta aos problemas colocados. Sendo assim, após o utilizador do sistema ter definido o tipo de problema de optimização que pretende resolver, com a ajuda providenciada pelo sistema, seguidamente terá de escolher um de entre os vários programas disponíveis para resolver o problema em questão. Esta ajuda inclui a apresentação de uma árvore relativa aos tipos de optimização, que inclui subtipos de optimização, nomeadamente optimização contínua, com ou sem restrições e optimização discreta, através de programação inteira, eventualmente combinada com programação estocástica. Subsequentemente, através desta árvore é possível desencadear o processo de pesquisa e aceder aos diferentes tipos de programas disponibilizados pelo sistema, nomeadamente implementações de métodos do tipo branch and bound, entre outros.

Em função do tipo de programa escolhido é apresentada a forma requerida para a introdução dos dados do problema a resolver, através do envio de um ficheiro, num determinado formato especificado, de acordo com o tipo de linguagem em que o programa foi implementado, entre outras características particulares de cada programa. Cada programa fornece exemplos de problemas e informação adicional sobre a sua utilização.

O BBN's Vishnu scheduling system tendo características um tanto específicas, também pode ser usado para resolver problemas de escalonamento da produção, via web.

O FortMP é um sistema essencialmente vocacionado para dar resposta a problemas que ocorrem no âmbito da investigação operacional, de modo a resolver problemas do tipo, de transporte, modelação económica, sistemas e redes de energia, usando essencialmente técnicas de programação matemática. Grande parte dos problemas industriais ou de investigação que envolvam optimização linear ou discreta podem ser resolvidos através deste sistema. A utilidade deste sistema para a resolução de problemas de escalonamento da produção é limitada.

O e-OCEA é um sistema mais amplo e também mais próximo dos objectivos propostos com este trabalho, tendo em vista ajudar na identificação de problemas de escalonamento e

ajudar no desenvolvimento de novos métodos, além de conduzir fóruns de discussão através da Internet. Contudo, um requisito deste sistema é o de apenas considerar elementos (métodos, conjuntos de dados, programas e módulos que sejam compatíveis com o sistema e-OCEA).

Relativamente aos dados e aos programas isto significa que na BD eles estão descritos em ficheiros com um mesmo formato. Relativamente aos métodos, esta compatibilidade com e-OCEA requer que estes sejam capazes de ler dados e devolver programas no formato de ficheiro usado pelo projecto e-OCEA. Os requisitos para os módulos são da mesma natureza.

Este requisito de compatibilidade com o formato de ficheiro (um ficheiro de texto) para conjuntos de dados, programas, etc. torna necessário desenvolver e disponibilizar, em diferentes linguagens de programação, códigos para ler e escrever em ficheiros contendo conjuntos de dados e programas. Os utilizadores do sistema são convidados a reutilizar estes códigos se estiverem interessados em desenvolver uma heurística ou um módulo que seja compatível com o e-OCEA, de modo a poder aí ser integrado.

Este portal oferece vários módulos que providenciam serviços gratuitos. Cada módulo ajuda o utilizador numa determinada fase do "ciclo de vida" de um problema de escalonamento. Desta forma, existem módulos para diversos fins, nomeadamente para permitir ao utilizador modelar graficamente o seu problema e para ajudar a gerar o código associado a métodos desenvolvidos. Existe também um módulo para avaliação computacional e comparação dos métodos a testar, com base em dados introduzidos pelo utilizador. Existe ainda um módulo que permite a visualização e edição de programas na forma de diagramas de Gantt.

A visualização e consulta de informação é baseada numa classificação de problemas que um processo muito simples e imediato. Uma árvore representa esta classificação, para um conjunto de classes pré-definidas: percorre-se então à árvore de modo a seleccionar um tipo de problema específico que se aproxime o mais possível daquele que o utilizador pretenda resolver. Seguidamente, surgirá uma lista de métodos, conjuntos de dados ou referências que estejam disponíveis na base de dados do sistema, correspondentes ao tipo de problema seleccionado. A filosofia subjacente a este site de web consiste em aceder sempre à informação graças a pontos de entrada pré-definidos, que estejam relacionados com os problemas práticos a resolver, que se espera possam ser úteis para a prática industrial mais comum.

O LekiNET, anteriormente mencionado, é um sistema que também possui algumas similaridades com este trabalho mas é, contudo, essencialmente vocacionado para a obtenção de escolhas que sejam vantajosas do ponto de vista económico, de agentes de escalonamento para a resolução de problemas.

Neste sistema os autores propõem um esquema migratório para transformar sistemas de escalonamento autónomos existentes em agentes de escalonamento da Internet, que possam comunicar entre si e resolver problemas para além das suas capacidades individuais. O enquadramento do sistema trata cada sistema como um agente e constrói as relações entre esses sistemas. Sendo assim, torna-se necessária a existência de "wrappers" que têm de ser especificamente projectados para cada sistema particular.

Com este trabalho pretende-se avançar com uma proposta de um projecto, através do qual seja possível dar continuidade ao trabalho levado a cabo com esta tese, contribuindo para a construção de uma plataforma de web que permita, de um modo relativamente fácil, disponibilizar métodos para a resolução de problemas de escalonamento da produção industrial, requerida por qualquer utilizador no espaço cibernético global.

O objectivo consiste em permitir que qualquer método, que esteja acessível através da Internet ou de uma outra rede qualquer possa ser usado, para a resolução de problemas colocados pelos utilizadores. A associação de problemas de escalonamento a métodos de resolução é feita usando a informação disponível acerca de problemas e métodos.

Da pesquisa de sistemas de escalonamento efectuada não foram encontrados sistemas com uma arquitectura, abordagem e/ ou objectivos similares aos que se propõem atingir nesta tese, isto é, orientados para a resolução de uma ampla variedade de problemas de escalonamento da produção industrial, com base numa base de conhecimento distribuída, flexível e facilmente actualizável, que permita providenciar um serviço de apoio ao escalonamento da produção aos seus utilizadores.

2.7 Considerações finais

Para concluir este capítulo, no que se refere aos métodos de escalonamento da produção, pode dizer-se que, métodos muito gerais, tendo um vasto campo de aplicação em geral, são normalmente pobres no que se refere ao desempenho. Métodos específicos adequados a um determinado problema atingem uma qualidade geralmente bastante maior, mas com utilização restrita noutros domínios de problemas. As estratégias de pesquisa local caem, de certo modo, entre estes dois extremos, onde os algoritmos genéticos e as redes neuronais

estão normalmente associados à primeira categoria, enquanto que a pesquisa *tabu* ou os métodos de *simulated annealing* são referidos como exemplos da segunda categoria (Blazewicz, 1996). Estes métodos podem ser vistos como ferramentas para pesquisar um espaço de alternativas possíveis, no sentido de encontrar uma boa solução, geralmente dentro de limites de tempo aceitáveis. Devido à dificuldade de encontrar algoritmos de optimização suficientemente rápidos são requeridas técnicas para a localização rápida de soluções de elevada qualidade em espaços de pesquisa amplos e complexos e sem qualquer garantia de optimabilidade das soluções obtidas. Quando existe conhecimento suficiente disponível é possível, frequentemente, explorar a capacidade de inferência desse conhecimento, no sentido de introduzir estratégias de procura específica do problema, capaz de servir de suporte para encontrar soluções de mais elevada qualidade. Sem um tal conhecimento, à partida, ou em casos em que soluções próximas do óptimo são indispensáveis, ter-se-á de acumular informação acerca do problema, de uma forma dinâmica, durante o processo de pesquisa.

Abordagens clássicas, relacionadas com espaços de pesquisa combinatorial, acerca dos quais existe pouco conhecimento disponível, não permitem geralmente "aprender" como obter um óptimo local. Por exemplo, considere-se a procura aleatória, neste caso a pesquisa pode ser mais eficiente se o espaço de procura for razoavelmente denso, com soluções aceitáveis, de tal modo que a probabilidade de encontrar uma solução seja elevada. Contudo, na maior parte dos casos, encontrar uma solução aceitável, num intervalo de tempo razoável é impossível, porque a pesquisa aleatória não usa qualquer conhecimento gerado durante o processo de pesquisa, no sentido de melhorar o seu desempenho.

Combinando o *hill-climbing* assim como a amostragem aleatória de um modo criativo e introduzindo conceitos de aprendizagem e memória poderão ser superadas as deficiências anteriormente mencionadas para os métodos de pesquisa local. As estratégias de *local search based learning* são conhecidas, por exemplo, sob a designação de pesquisa tabu e algoritmos genéticos. Estas técnicas providenciam estratégias gerais para a resolução de problemas, incorporando e explorando conhecimento específico do problema, capaz mesmo de explorar espaços de procura contendo um número exponencialmente crescente de óptimos locais, relativamente aos parâmetros de definição dos problemas.

Existem ainda um conjunto de outros tipos de métodos, nomeadamente os métodos baseados em redes de Petri, que têm vindo a ser cada vez mais desenvolvidos.

No intuito de lidar com a natureza imprecisa de dados e de julgamentos de decisores, aquando da resolução de problemas de escalonamento da produção, as técnicas de optimização têm vindo a ser combinadas com abordagens e conceitos da lógica difusa, nomeadamente da teoria de conjuntos difusos (Ribeiro, 1999).

Bellman e Zadeh (Petrovic, 2004) introduziram uma abordagem para resolver problemas de optimização de objectivo único e multi-objectivo em ambientes difusos. Eles definiram uma única função de decisão que agrega graus de satisfação atingidos relativamente a ambos, objectivos e restrições difusos. Neste contexto, o operador *min* (*minimum*) tem sido mais frequentemente usado como um operador de agregação (Petrovic, 2004).

Relativamente aos sistemas de apoio ao escalonamento da produção pode concluir-se que, embora a quantidade e variedade de tipos de sistemas de apoio ao escalonamento da produção tenha vindo a aumentar significativamente e estes sistemas estarem também, cada vez mais, a ser disponibilizados via Internet, ainda se verifica, porém, que tais sistemas continuam a ser essencialmente vocacionados para a resolução de um conjunto mais ou menos restrito de tipos de problemas, com grande incidência em problemas que ocorrem em sistemas do tipo oficinas de fabrico e sistemas flexíveis de produção, com base num conjunto também mais ou menos restrito de métodos para a sua resolução, isto é, sistemas com uma abrangência restrita para a resolução de problemas mais ou menos relacionados, com base num conjunto de métodos com algum grau de afinidade entre si, nomeadamente métodos que se baseiam em técnicas de pesquisa de soluções similares, princípios de funcionamento proximamente relacionados ou métodos heurísticos simples. Além disso, normalmente, os problemas abordados consideram os critérios de optimização mais comuns ou usuais ou mesmo um único critério ou apenas alguns critérios de optimização relacionadas e, tipicamente, um conjunto restrito de métodos que permitem considerar também um conjunto restrito e normalmente estático (não expansível) de critérios e de problemas possíveis de resolver. Trata-se, portanto, geralmente, de um conjunto mais ou menos restrito de métodos ou um único método, desenvolvido para dar resposta a esse conjunto também restrito de problemas, muitas vezes pertencentes a uma única classe específica de problemas ou a algumas, poucas classes de problemas relacionados.

Como se referiu no capítulo 1, neste trabalho o que se pretende é precisamente contrariar esta lógica, com a proposta de um sistema abrangente, ou seja, um sistema com uma arquitectura tal que permita disponibilizar um conjunto vasto e diversificado de métodos, para a resolução de um conjunto também amplo de problemas de escalonamento da

produção diversificados, que possam ocorrer nos mais diversos sectores industriais, em diferentes sistemas de produção e de modo a permitir considerar um amplo e expansível conjunto de critérios de optimização. Ou seja, um sistema que permita dar resposta à medida de diversificadas necessidades industriais concretas, típicas de cada contexto particular em que os problemas de escalonamento da produção ocorram.

Um sistema desta natureza pode incluir métodos baseados nas mais diversas abordagens e técnicas de pesquisa de soluções, que podem estar implementados nas mais diversas linguagens de programação e correr nos mais diversos processadores, com diferentes plataformas/ sistemas operativos e diversificadas tecnologias de suporte subjacentes, que estejam acessíveis via Internet ou de uma qualquer rede, intranet, através de um repositório distribuído de conhecimento de escalonamento.

3 Problemas de Escalonamento da Produção

3.1 Introdução

O escalonamento da produção, frequentemente também referido como programação da produção, razão pela qual usaremos nesta tese os dois termos de forma indistinta, visa a programação da execução de um conjunto de trabalhos num conjunto de unidades de produção, durante um determinado intervalo de tempo.

O resultado do escalonamento da produção é um programa de produção. Este pode ser visto a vários níveis hierárquicos de decisão, desde o estratégico, referido como programa de produção agregada, ao operacional, expresso em programas detalhados de lançamento e/ ou de fabrico no espaço fabril, passando pelo táctico, do que um programa director de produção é paradigma. Neste trabalho de investigação o escalonamento é equacionado na óptica do escalonamento operacional ou fabril e principalmente naquela que envolve a afectação, despacho e sequenciamento, i.e. na óptica do que vulgarmente se costuma referir como escalonamento detalhado ou de curta duração. Este está particularmente associado à função de controlo da actividade de produção. A afectação é uma função do escalonamento a resolver quando existem processadores alternativos que podem ser usados para realizar, i.e. executar, processar ou transformar um trabalho ou tarefa. O despacho escolhe e encaminha os trabalho para execução, de forma iterativa, à medida que os processadores vão terminando a execução de tarefas e ficando disponíveis. Normalmente as tarefas, i.e. os trabalhos, encontram-se em espera, no sistema de produção ou na fila do centro de trabalho ou posto de trabalho onde é transformado. Um posto de trabalho, ou simplesmente posto, é o local onde o trabalho sofre um processo de transformação, sendo constituído por um ou mais processadores usados simultaneamente no processamento de uma tarefa. Um centro de trabalho refere-se a um conjunto de processadores agrupados que são organizados em postos de trabalho. O sequenciamento é uma função alternativa ao despacho, i.e. quando uma existe a outra não, identificando antecipadamente a ordem de execução dos trabalhos em cada posto, por vezes resolvido integradamente com o problema de afectação. Trata-se

de uma função de ordenação de tarefas para execução em cada posto. Frequentemente o escalonamento é visto como a acção integrada de afectação e sequenciamento. No entanto, em muitas situações o problema do escalonamento reduz-se ao de sequenciamento ou despacho. Tal acontece quando o problema de afectação não se põe, por não haver postos alternativos de execução das tarefas a realizar, ou quando o problema de afectação já foi resolvido. O sequenciamento pode, também, ser referido por programação sequencial ou mesmo sequenciação.

Subjacente a cada programa de produção pode estar associada a calendarização das tarefas, definindo-se desta forma e claramente o instante de começo e término de cada tarefa a realizar em cada posto. Isto só é possível se tanto o problema de afectação como o de sequenciamento estiverem resolvidos.

Um problema de escalonamento resulta, portanto, da necessidade de obter um programam de produção para a execução de um número de trabalhos num determinado ambiente de produção, caracterizado pela existência de uma variedade de recursos ou meios de produção principais, identificados nesta tese como processadores, tais como máquinas ou postos individuais de fabricação ou montagem, e de meios ou recursos de produção auxiliares, tais como ferramentas, dispositivos de fixação e manipuladores e mesmo operadores humanos quando apenas auxiliam o recurso principal a executar a sua tarefa. Isto acontece em todos os processadores mecanizados e/ ou semi-automáticos. Nos postos manuais de produção o recurso principal é operador humano.

Portanto, para uma verdadeira compreensão da complexidade e extensão da problemática de escalonamento é importante conhecer e caracterizar, nos aspectos relevantes ao escalonamento:

- Os trabalhos ou tarefas
- Os processadores
- Os meios auxiliares
- Os ambientes de escalonamento e de produção
- Os critérios de optimização

e, ainda, saber como é que estes elementos se relacionam e contribuem para execução de um conjunto de trabalhos, cada um dos quais sujeito a um plano operatório (Carmo-Silva, 2005). Este plano que pode considerar-se como outro elemento fundamental do processo de

escalonamento, caracteriza os trabalhos no âmbito restrito do sistema de produção disponível, definindo as operações, necessárias à realização do trabalho e identificando que processadores e meios auxiliares que são necessários para a execução de cada operação, assim como as restrições à ordem de execução de cada uma.

O último elemento fundamental do escalonamento é o objectivo a atingir com o programa de produção que se pretende, i.e. a solução obtida pelo método ou processo de escalonamento usado. As soluções que se podem obter são, portanto, sempre dependentes dos objectivos de optimização. Em geral podemos dizer que o escalonamento permite planear o uso eficiente dos recursos de produção baseado em objectivos, expressos através do objectivo de optimização definido. Este é função de medidas de avaliação da qualidade das soluções, referidas como medidas de desempenho. Embora as soluções obtidas, i.e. os programas de produção, possam ser avaliadas em função de muitas medidas de desempenho, aquela a que o processo de escalonamento se sujeita, i.e. que define o objectivo de optimização, é geralmente referida como critério de optimização. Quando mais que uma tal medida é usada na definição do objectivo de optimização diz-se que estamos perante optimização multi-critério (Ribeiro, 2000).

Nas secções seguintes deste capítulo estuda-se cada um dos elementos referidos, i.e., os trabalhos ou tarefas, os recursos principais e auxiliares, o ambiente e os critérios de optimização, por serem essenciais à caracterização dos problemas diversificados de escalonamento que se encontram na teoria e na prática. Por fim, faz-se uma breve referência à problemática da complexidade e diversidade de problemas de escalonamento e à forma de lidar, na prática, com estas questões.

3.2 Trabalhos, processadores e recursos auxiliares

O escalonamento ocorre num vasto leque de actividades e envolve sempre a interacção entre diferentes tipos de processadores, necessários num determinado período de tempo, para realizar um conjunto de trabalhos.

Em geral, os problemas de escalonamento são caracterizados por quatro conjuntos de dados ou variáveis: O conjunto $\mathsf{T}=\{T_1, T_2, \ldots, T_n\}$ de $\underline{n}$ trabalhos T_j, o conjunto $\mathsf{O}=\{O_1, O_2, \ldots, O_{nj}\}$ de $\underline{nj}$ operações O_{ji}, o conjunto $\mathsf{M}=\{M_1, M_2, \ldots, M_m\}$ de $\underline{m}$ processadores M_i e o conjunto $\mathsf{R}=\{R_1, R_2, \ldots, R_{ms}\}$ de $\underline{ms}$ recursos auxiliares R_k. O escalonamento, de um modo geral, significa atribuir processadores de M e, possivelmente, recursos de R a operações de

O, ou vice versa, no intuito de realizar todos os trabalhos de T, sob as condições e restrições impostas (Pinedo, 2002).

Existem outros conceitos fundamentais para a caracterização dos problemas de escalonamento da produção, nomeadamente, dados temporais, como por exemplo, prazos ou datas de entrega, ou tempos de processamento, que cada trabalho ou operação requer para a sua execução, durante um determinado período. Os processadores, por sua vez, também podem ser de natureza muito diversa, incluindo os próprios operadores, ferramentas e dispositivos diversificados, nomeadamente transportadores e também possuem dados temporais associados, tais como, tempos de transporte, esperas ou tempos de inactividade.

Estes conceitos serão aqui apresentados em dois blocos fundamentais. Num primeiro bloco serão apresentadas definições relativas aos trabalhos e às operações e no segundo bloco as definições respeitantes aos processadores e aos recursos do sistema de produção.

3.2.1 Caracterização e nomenclatura de representação de trabalhos

A seguir apresentaram-se definições básicas relativas aos trabalhos e às suas características principais.

Definição 3.1 – Trabalho (Tj, com $1 \leq j \leq n$ e $n \geq 1$): consiste num conjunto de operações $O_{ji} \in O=\{O_{11}, O_{21}, \ldots, O_{ji}\}$ $(1 \leq i \leq n_j$ e $n_j \geq 1)$ $O_{j1}, O_{j2}, \ldots O_{j,nj}$. necessárias à obtenção de um determinado produto. No caso de $n_j=1$, para algum $T_j \in T=\{T_1, T_2, \ldots, T_n\}$, $O_{ji}=O_{j1}=O_j=1$, então trabalho será usado como sinónimo de operação, ou seja $T_j=O_j$.

Entende-se, portanto, por trabalho, um conjunto de operações, de transformação de uma entidade, necessárias à obtenção de um determinado produto, bem ou serviço. Produtos podem ser bens, obtidos através dos processos de transformação inerentes aos sistemas de produção industrial, deste modo os termos "trabalhos" e "operações" serão usados para identificar as entidades cuja transformação se pretende escalonar.

Definição 3.2 - Trabalho composto ($comp_j$): é um trabalho constituído por 2 ou mais componentes ou subprodutos, cada um consistindo em trabalhos, isto é, sub-trabalhos, que se decompõem em operações. Estas operações, dos vários sub-trabalhos ou componentes que integram o trabalho composto, podem ser processadas em simultâneo ou de uma forma sequencial e encadeada, em processadores diferentes ou similares, aquando da fase de fabrico dos seus componentes e das operações de montagem (Ribeiro, 2002).

Definição 3.3 – Lote (batch): é um conjunto de trabalhos que são processados num mesmo processador, ou conjunto de processadores que integram um determinado sistema de produção e são, por isso, agrupados formando um conjunto de trabalhos que são processados conjuntamente.

Numa visão alargada um lote pode ser entendido como um conjunto de instâncias de um mesmo trabalho/ operação ou de trabalhos/ operações semelhantes, como uma família de produtos, a serem lançados simultânea ou encadeadamente em fabrico, num sistema de produção, para serem processados. Quando se faz a programação de lotes de trabalhos considera-se aqui o trabalho representado por (l, j), que indica o trabalho j do lote l.

No caso da produção de um lote de trabalhos ser simultânea no processador estar-se-á perante um problema de escalonamento em que, adicionalmente, se considera a existência de processadores multi-item.

No caso de se tratar de processamento não simultâneo de lotes, mas sequencial, o instante de conclusão C de um lote l é o instante de conclusão de todos os trabalhos do lote, que poderá ser definido pelo instante de conclusão do último trabalho desse lote.

De um modo genérico, um lote poderá ter um tamanho que varia desde um único trabalho até n trabalhos, por lote ou n_l, no caso de o tamanho dos lotes de trabalhos ser variável de lote para lote. No primeiro caso tratar-se-á de um lote de tamanho unitário ou de um item, o que equivale de facto a uma situação de "não existência de produção em lotes".

Num problema com dimensionamento de lotes (*lot sizing*), normalmente existe um tempo de preparação para cada lote e o escalonamento da produção consistirá em programar lotes em vez de trabalhos individuais, podendo ocorrer diferentes cenários, como se descreve a seguir.

Casos particulares de produção em lotes

Lotes de trabalhos iguais simples não relacionados

Todos os trabalhos do lote são iguais e associados a produtos simples ou peças. Por exemplo, produzir 10 unidades (trabalhos) da mesma peça.

Os trabalhos do lote são vistos como não relacionados por não terem uma dependência tecnológica, i.e., a execução de um não depende tecnologicamente da execução de outro. O seu agrupamento em lotes tem razões puramente operacionais ou organizacionais com vista a economia produtiva.

Lotes de trabalhos diferentes simples não relacionados

Vários artigos simples, que podem ser diferentes, são agrupados num lote para fabrico conjunto, normalmente para explorar vantagens da tecnologia de grupo em linhas ou células de produção. Os trabalhos do lote são vistos como não relacionados por não terem uma dependência tecnológica i.e., a execução de um não depende tecnologicamente da execução de outro. O seu agrupamento no mesmo lote tem razões puramente operacionais com vista a economias produtivas, por exemplo, economias de preparação dos processadores ou *setup*. É exemplo o chamado *"group scheduling"* (Jordan, 1996) baseado em de famílias de artigos.

Lotes de trabalhos diferentes relacionados – artigo composto

Vários trabalhos diferentes, para produção de artigo composto são agrupados num lote por envolverem a fabricação de todos os componentes e a sua montagem para a obtenção de um artigo (trabalho) composto final único.

Os trabalhos do lote são vistos como relacionados por terem uma dependência tecnológica i.e., a execução de uns depende tecnologicamente da execução de outros.

Por exemplo, sem se obterem dois componentes eles não podem ser montados. O seu agrupamento no mesmo lote justifica-se por fazer parte de um todo integrado visto, após processamento, como **um artigo único**.

Estes lotes estão associados à fabricação de um artigo ou trabalho composto por vários componentes, normalmente representado por uma lista de materiais (*Bill of Materials*, *BOM*) em árvore invertida, como é típico em sistemas de determinação de necessidades de materiais (*Materials Requirement Planning*, *MRP*).

Lotes de trabalhos iguais compostos – lotes de artigos compostos

Todos os trabalhos do lote são iguais e exercem-se sobre produtos (trabalhos) compostos. i.e. produtos que incluem dois ou mais produtos simples que são montados. Por exemplo, produzir 10 bicicletas do mesmo modelo.

Os trabalhos do lote podem ser vistos como não relacionados por não terem uma dependência tecnológica, i.e., a execução de um (uma bicicleta, por exemplo) não depende tecnologicamente da execução de outro (por exemplo, outra bicicleta igual). No entanto, a coordenação da produção de componentes com a sua montagem está relacionada.

O seu agrupamento em lotes tem razões puramente operacionais ou organizacionais com vista à eficácia e economia produtiva. Por exemplo, são produzidas juntamente por economias de escala, quer na fabricação de componentes quer na sua montagem, ao mesmo tempo que se pretende coordenar a sua fabricação conjunta.

Os trabalhos ou lotes de trabalhos têm subjacente um conjunto de características que podem surgir combinadas de diversas formas, dando origem a problemas de escalonamento mais ou menos diversos e complexos.

Um lote constituído por várias instâncias de trabalhos da mesma natureza conduz, por vezes, a um tipo de programa designado na literatura por *critical job schedule* (Morton, 1993), em que todas as operações do trabalho "mais crítico" são realizadas no sistema, depois o segundo mais crítico, e por aí em diante. A ideia principal consiste em ter uma atenção especial para com os trabalhos mais urgentes, de modo a que sejam executados primeiro, enquanto que os trabalhos menos urgentes ficam à espera que os processadores voltem a ficar disponíveis, para serem executados.

A seguir apresenta-se uma descrição das principais características dos trabalhos.

Definição 3.4 - Família de trabalhos (fam): consiste, segundo Jordan (1996), num agrupamento de trabalhos com semelhanças, normalmente de natureza processual, de modo a reduzir os custos de produção, nomeadamente no que se refere a custo de preparação dos processadores. Trata-se, portanto, de uma produção em lotes de trabalhos simples relacionados ou não.

As famílias de trabalhos resultam de um processo de agregação, que consiste em associar ou juntar trabalhos "semelhantes" no intuito de formar grupos de trabalhos e, deste modo, transformar um problema de sequenciamento tradicional num problema expandido do tipo programação com agrupamento de trabalhos ou *batch sequencing problem (BSP)* (Jordan, 1996).

O problema fundamental que se coloca a este nível de planeamento está relacionado, segundo Jordan (1996), com o equilíbrio entre a produção e a procura. Este problema remete para duas decisões independentes a serem tomadas: como agrupar os trabalhos em lotes e como sequenciá-los. Jordan afirma ainda que é necessário chegar a uma situação de compromisso porque, em geral, nem se devem agrupar todos os trabalhos de uma família para serem produzidos num mesmo lote, nem produzir cada trabalho em lotes individuais. No primeiro caso, seriam violados os prazos de produção e/ou mantidos elevados níveis de

existências e no último caso, seria desperdiçado demasiado tempo na preparação do sistema de produção.

Definição 3.5 - Filosofia de chegadas (rj): descreve as "chegadas" dos lotes ou trabalhos ao sistema produtivo, isto é, indica o instante de tempo, r_l, r_j ou r_{ji} ao qual o lote l (L_l), o trabalho j (T_j) está disponível para ser processada no sistema (Blazewicz, 1996).

Esta filosofia pode variar desde uma determinada distribuição estatística mais ou menos complexa até à suposição de que, por exemplo, todos os trabalhos chegam ou estão disponíveis no sistema, simultaneamente, num determinado instante, para serem processados. Diz-se, neste caso que estamos em ambiente estático e que no primeiro caso estamos em ambiente dinâmico.

Se, por exemplo, os tempos de chegada são os mesmos para todos os trabalhos T_j de T então pode assumir-se que, $r_j = 0$, caso contrário, os r_j variam de trabalho para trabalho.

Os instantes de chegada são obviamente conhecidos em sistemas de escalonamento que funcionam em modo *off-line*, isto é, em sistemas em que os tempos de chegada dos trabalhos ao sistema são conhecidos à *priori* e em sistemas de controlo em que são adquiridas amostras de tempos, através de sensores, em instantes de tempo fixos, previamente estipulados.

Definição 3.6 - Interrupção[1] (pmtn): significa que o processamento de um lote, trabalho ou operação pode ser interrompido e posteriormente retomado, no mesmo ou noutro processador.

Um programa é designado um "programa com interrupção", segundo Brucker (1995), se cada lote, trabalho ou operação pode ser interrompido(a), em qualquer instante de tempo, um número finito de vezes[2] e reiniciado(a), posteriormente.

Morton (1993) distingue vários tipos de interrupção, dos quais se poderão salientar os seguintes:

"Interrupção complexa"[3], um tipo de interrupção em que, ao interromper um lote, trabalho ou operação se incorre numa determinada penalização. Esta penalização pode ser devida à necessidade de voltar a efectuar a preparação do processador para retomar o processamento de um lote, trabalho ou operação ou mesmo devido à necessidade de reiniciar todo o

[1] Do inglês, preemption.
[2] Esta condição só é imposta por considerações de ordem prática.
[3] *Complex preemption*.

processamento, com as perdas de produtividade que daí advêm. Num caso mais extremo poderá dar-se o caso de haver uma deterioração de material e/ ou de ferramentas ou equipamentos, resultante dessa interrupção. A outra, a "interrupção livre"[1], é um tipo de interrupção em que se permite o processamento simultâneo de diferentes componentes ou instâncias de um trabalho de um lote em processadores diferentes. Este tipo de "interrupção" está associado à partição de lotes.

Definição 3.7 – Prioridades (wj): o "peso" ou prioridade (wj) expressa a urgência ou importância relativa do lote ou do trabalho. Trata-se de um parâmetro que permite distinguir a importância, urgência ou prioridade dos diferentes lotes, trabalhos ou operações e funciona, portanto, como uma espécie de peso, que permite avaliar essa 'importância' ou prioridade de cada lote, trabalho ou operação (Brucker, 1995).

Definição 3.8 - Restrições de precedência (prec): são restrições que podem existir entre as operações constituintes de um determinado trabalho, ou entre trabalhos e são, normalmente, representadas por um grafo direccionado acíclico (Brucker, 1995). As restrições de precedência entre operações são definidas, no conjunto O, por $O_i \prec O_t$, significando que o processamento da operação O_i tem de estar concluído antes de a operação O_t poder ser iniciada. Por outras palavras, é definida, no conjunto O, uma relação de precedências $\prec$. As operações do conjunto O são designadas dependentes se a ordem de execução de pelo menos duas operações em O é restringida por esta relação. Caso contrário, as operações são designadas independentes (Brucker, 1995).

Um conjunto de operações com restrições de precedência pode ser representado na forma de um grafo direccionado em que os nós correspondem a trabalhos ou operações e os arcos a restrições de precedência (um grafo com os trabalhos ou operações representados nos nós é um grafo do tipo *task-on-node graph (TON)* ou *activity-on-node (AON)*. Um exemplo de um conjunto de trabalhos dependentes é mostrado na Figura 3.15 (a), onde os nós são representados por T_j/ t_j.

Convém referir que geralmente, excepto por exemplo, em sistemas abertos, as operações que constituem um trabalho são dependentes, mas os próprios trabalhos podem também ser dependentes ou independentes, isto é, há casos em que a execução de um trabalho depende da realização prévia de outro(s).

[1] *Free-preemption* ou *splitting*.

Existe outra forma de representar as dependências entre os trabalhos ou as operações, que é vantajosa em determinadas circunstâncias. A designada representação em redes ou grafos com os trabalhos ou operações representados nos arcos, onde os arcos representam, portanto, os trabalhos/ operações e os nós os eventos temporais. Um grafo especial deste tipo é designado *uniconnected activity network (uan)*, que é definido como um grafo em que cada dois nós estão conectados por um ou mais caminhos orientados.

Para cada grafo de precedências poder-se-á construir uma correspondente rede de trabalhos (e vice versa), usando trabalhos fictícios com comprimento nulo. A correspondente rede de trabalhos para o grafo de precedências da Figura 3.1 (a) está representada na Figura 3.1 (b).

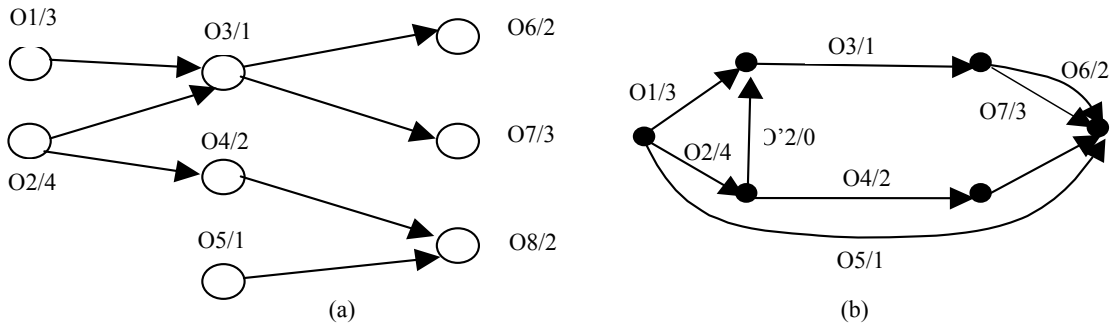

Figura 3.1 – (a) Grafo de precedências com os trabalhos nos nós e (b) nos arcos.

A operação O_{ji} considera-se disponível no instante de tempo t se $r_{ji} \leq t$ e todos os seus predecessores (relativamente às restrições de precedência) estejam concluídos no instante t.

Quando se impõem precedências nos trabalhos ou operações existe um conjunto de trabalhos/ operações que terão de estar concluídos(as) para que se possa dar início a cada trabalho/ operação seguinte.

Brucker (1995) afirma que quando estas restrições existem entre trabalhos estes são considerados não independentes e as dependências serão representadas por um grafo orientado acíclico G = (V, A), onde, V={1, …, n} correspondente aos trabalhos (j, s) $\in$ A se $T_j \rightarrow T_s$.

Existem outras formas de representar este tipo de restrições, nomeadamente, através de caminhos[1], ou de árvores[2] (Blazewicz, 1996).

[1] *Chains.*
[2] *Trees.*

Definição 3.9 - Restrições nos tempos de processamento (tji): as restrições nos tempos de processamento (tji) são restrições que indicam os valores que as variáveis dos tempos de processamento podem assumir para as diferentes operações dos vários trabalhos (Brucker, 1995).

O vector dos tempos de processamento $t_{ik} = [t_{1k}, t_{2k}, \ldots, t_{nj,m}]^T$, é um vector onde t_{ik} é o tempo necessário pelo processador M_k , para processar a operação O_i, em que, $1 \leq k \leq m$ e $1 \leq i \leq n_j$, para cada trabalho T_j, em que $1 \leq j \leq n$.

No caso do escalonamento o vector dos tempos de processamento descreve os requisitos de processamento para uma determinada operação pertencente a um determinado trabalho, isto é, para o trabalho T_j verificar-se-á que, $t_{ji} = [t_{j1}, t_{j2}, \ldots, t_{n,nj}]^T$, onde t_{ji} representa o tempo de processamento de O_{ji}, ou seja, da operação O_i, do trabalho T_j. Deste modo, cada trabalho T_j é processado num intervalo de tempo que se inicia com o seu instante de chegada ao sistema de produção, isto é, no intervalo $[r_j, \infty[$, (Blazewicz, 1996).

Numa abordagem determinística, quando num programa de produção se pretendem cumprir prazos de produção, a abordagem (quando o tempo de processamento dos trabalhos não é conhecido) consiste em resolver o problema assumindo limites superiores para esses tempos. Por outro lado, uma outra abordagem possível consiste em considerar valores médios para os tempos de processamento.

Normalmente, os tempos de processamento dos trabalhos ou operações estão sujeitos a determinadas condições e Brucker (1995) apresenta os seguintes casos:

$t_{ji} = 1 \rightarrow$ quando todas as operações i do trabalho j têm tempo de processamento unitário.

$t_i=1 \rightarrow$ quando o trabalho j tem uma única operação i e o seu tempo de processamento é unitário.

$t_j \in \{1, 2, \ldots, p\} \rightarrow$ quando os tempos de processamento dos trabalhos variam num conjunto de valores possíveis.

Blazewicz (1996) considera ainda os seguintes casos:

$t_j = p \rightarrow$ quando todos os trabalhos j têm um tempo de processamento constante e igual a p unidades.

$t_{inf} \leq t_j \leq t_{sup} \rightarrow$ quando todos os trabalhos j têm tempos de processamento no intervalo [tinf, tsup]

Definição 3.10 - Data ou prazo de entrega ou de conclusão (dj): uma data ou prazo de entrega ou de conclusão, dj, segundo Brucker (1995), especifica um instante de tempo, em termos de data de entrega ou instante de conclusão de um lote, de um trabalho ou de uma operação, respectivamente, em que o processamento desse lote L_l, trabalho T_j ou operação O_{ji} deverá estar concluído.

Geralmente são definidas funções de penalização de acordo com os atrasos no processamento destas entidades em relação a estes prazos.

Segundo Jordan (1996), em algumas abordagens os prazos de produção poderão ser violados e aí designar-se-ão, preferencialmente, prazos de entrega em vez de prazos de produção. Nesse caso, considera-se frequentemente a minimização do atraso máximo como objectivo do problema de escalonamento, visto que este será usado como um critério equivalente à avaliação da perda de "boa vontade" dos clientes perante encomendas atrasadas. Esta abordagem foca a sua atenção preferencialmente no mercado do que no processo de fabrico.

Blazewicz (1996), afirma que são considerados prazos, em problemas de escalonamento, quando são impostas datas de entrega ou de conclusão, ou quando estes estão indirectamente reflectidos no problema, através da sua inclusão numa critério de optimização do sistema produtivo, aquando da programação de um conjunto de lotes, trabalhos ou operações.

Definição 3.11 - Operação multi-processador (mpti ou mptj): Segundo Brucker (1995) uma operação i (ou trabalho j) multi-processador é mpt[1]: $mpt \in \{(mpt, \{M_1, \ldots, M_k\}) \mid 2 \leq k \leq r_i\}$. Isto é, uma operação i poderá requerer mais do que um processador k, simultaneamente, para ser processada.

Para o processamento da operações dos trabalhos existe, por vezes, a necessidade de outros recursos, além dos principais ou processadores propriamente ditos, que são os designados recursos auxiliares.

Por exemplo, durante um determinado intervalo de tempo a operação O_{ji} poderá necessitar, simultaneamente de: processador w + processador x + ferramenta y + robot z.

[1] *Multi-processor task.*

3.2.2 Caracterização e nomenclatura de representação de processadores e de recursos

De um modo similar ao que anteriormente se ilustrou para os trabalhos, os processadores e os recursos auxiliares de produção têm também subjacente um conjunto de características que se podem combinar de diversas formas nos problemas de escalonamento, conferindo a estes complexidade e diversidade diversas. A seguir descrevem-se essas características, que serão posteriormente usadas na nomenclatura de classificação dos problemas apresentada no capítulo 4.

Definição 3.12 – Processador (Mk): é um recurso de produção principal, para a realização de um trabalho, que pode ser um qualquer posto de trabalho manual e/ou mecanizado ou automático.

Definição 3.13 - Recurso auxiliar (auxk ou Rk): é qualquer recurso necessário, durante um determinado intervalo de tempo, além dos processadores (recursos principais), que irão ser utilizados na execução dos trabalhos, para a realização completa das operações que o integram (Blazewicz, 1996).

Em ambientes de produção poderão considerar-se operadores, ferramentas, robôs e veículos de transporte, entre outros, como sendo recursos auxiliares, quando auxiliam os processadores a executar os trabalhos.

Daqui em diante, optar-se-á pela designação de processador para representar, genericamente, um recurso de produção principal e o termo recurso para expressar um recurso de produção auxiliar, para a realização dos trabalhos.

Definição 3.14 - Processador multi-função (mpmk[1]): é um processador que permite executar um vasto leque de operações de trabalhos diferentes podendo, no limite, processar qualquer operação de qualquer trabalho, uma vez equipado com a(s) ferramenta(s) adequada(s) (Brucker, 1995).

Definição 3.15 - Processador multi-item[2] (bmk): é um recurso de produção, nomeadamente, um processador, que tem a capacidade de processar ou manipular dois ou mais lotes, trabalhos ou operações simultaneamente e em que não existe, portanto, necessidade de preparação desse processador entre os vários lotes, trabalhos ou operações em questão (Morton, 1993).

[1] *Multi-purpose machine.*
[2] *Batch machine*.

Um exemplo de um processador multi-item é, por exemplo, um forno que faz a secagem de solda em várias placas de circuito impresso simultaneamente.

Definição 3.16 - Processador crítico (crtk): é uma processador considerado "mais importante" por ser um dos mais utilizados, que está permanentemente ou temporariamente em sobrecarga e, portanto, pode provocar a ocorrência de gargalos de estrangulamento e possivelmente até a paragem do sistema produtivo (Morton, 1993).

Os problemas de escalonamento em que se considera a existência deste tipo de processadores conduzem a programas de produção que são normalmente designados de *critical resource schedule* ou *bottleneck schedule*. Nestes programas há, geralmente, a preocupação de programar primeiro o processador crítico (*bottleneck*) e só depois os restantes processadores são programados em função deste. A ideia principal subjacente a tal procedimento centra-se no facto de que se espera que o uso eficiente do processador crítico ajude a descongestionar o sistema de produção (Morton, 1993).

Definição 3.17 - Processadores agregados (agreg): consistem num conjunto de dois ou mais processadores agrupados que constituem o sistema produtivo que pode ser visto como um único processador pelo facto de existir uma única fila de espera de lotes, trabalhos ou operações à entrada destes e não haver nenhuma decisão adicional a ser tomada a nível interno desse sistema ou subsistema agregado além da ordenação destes para serem processados (Morton, 1993).

Os problemas de processadores agregados são vulgarmente designados «problemas de recurso único». Nestes problemas as únicas decisões a considerar são, portanto, o sequenciamento e os instantes de libertação dos lotes, trabalhos ou operações, da fila de espera para o sistema. Podem existir muitas filas internas, no sistema, que são "escravos". Nessas filas o sequenciamento e despacho são processos efectuados, normalmente, através de disciplinas simples, tais como *FCFS*, mas não há necessidade de tomar quaisquer decisões activas internamente (Morton, 1993).

Definição 3.18 - Disponibilidade dos processadores (availk): é a característica dos problemas de escalonamento que descreve a disponibilidade temporal destes e dos recursos auxiliares (Morton, 1993).

Esta propriedade indica que os processadores poderão estar sempre disponíveis no tempo, logo a partir de um determinado instante ou possuir uma disponibilidade limitada ou condicionada. Neste último caso poderá recorrer-se a funções de distribuição para descrever a disponibilidade dos diferentes processadores e recursos auxiliares no sistema de produção, nomeadamente, num determinado intervalo de tempo, ou recorrer a qualquer outra condição mais complexa para definir esta disponibilidade.

A inactividade ou indisponibilidade "forçada" dos processadores e/ou dos recursos auxiliares consiste em permitir que estes possam permanecer inactivos, durante o período de funcionamento normal do sistema de produção em que se inserem, por exemplo, para estarem disponíveis para processar eventuais trabalhos urgentes que possam estar na eminência de chegar ao sistema (Morton, 1993).

Thomas Morton (1993) define os programas em que não se permite a inactividade forçada dos processadores de *simple dispatch schedule*, em que os processadores nunca são mantidos inactivos em antecipação à chegada de trabalhos urgentes.

Definição 3.19 – Preparação (sjk): consiste num processo de mudança[1] ou alteração do estado do processador k, para executar o trabalho j e depende das características tecnológicas do processador (Jordan, 1996).

Essa preparação do processador é feita à custa do consumo de uma determinada quantidade de "energia" expressa, normalmente, em termos de custo ou de tempo gasto nesse processo de preparação, sem contribuir para a produção efectiva, isto é, para o aumento do valor económico do produto.

Uma preparação pode, por exemplo, envolver uma operação de limpeza ou aquecimento de um processador, antes da produção propriamente dita poder ser iniciada. Após a preparação, o processador encontra-se num determinado estado de preparação que permite realizar um determinado lote, trabalho ou operação.

Muitas vezes, o esforço de preparação está dependente da ordem de lançamento dos trabalhos no sistema de produção. Um exemplo é a preparação de uma linha para pintar automóveis, de tal forma que, se a cor da tinta muda, a limpeza da linha requer esforços de preparação dependentes da mudança de cor efectuada.

[1] S*etup* ou *changeover*.

Geralmente assume-se que o estado de preparação de um determinado processador não se altera durante o processamento. Contudo, isto nem sempre se verifica. Um exemplo deste último caso ocorre quando o processo de transformação é um tratamento térmico e há necessidade de ir variando as condições térmicas. Neste caso o estado de preparação do processador altera-se durante a produção.

A preparação poderá ser omitida se dois trabalhos de uma mesma família forem produzidos de uma só vez, isto é, ambos os trabalhos estão num mesmo lote e são processados simultânea ou sequencialmente. Deste modo, a formação de grupos de trabalhos significa, agrupar ou juntar, trabalhos numa família. De um ponto de vista económico, isto providencia uma forma de realizar economias de escala. Sendo assim, quanto maior o tamanho do lote menor o tempo ou custo de preparação, contudo, tal redução nos tempos ou custos de preparação é, muitas vezes, acompanhada de um aumento dos custos de inventário, dada a necessidade de manter um nível mais elevado de existências, para satisfazer os requisitos dos em cursos de fabrico. Deste modo, para encontrar uma "boa solução" será importante chegar a uma situação de compromisso, entre dois extremos, a produção de lotes unitários e a produção de lotes "muito grandes".

Definição 3.20 - Armazém local (bufferk): é um espaço intermédio, entre dois processadores consecutivos e adjacentes, dedicado a um determinado processador ou partilhado por dois ou mais processadores (Morton, 1993).

Quando existe um armazém local, este faculta a acumulação de uma determinada quantidade de trabalhos, que aguardam, para prosseguir o seu processo de transformação pelo sistema de produção. Essa quantidade correspondente à capacidade de armazenamento do armazém local em causa e ocorre, por exemplo, durante um determinado intervalo de tempo $[t_{j,inf}; t_{j,sup}]$, para o trabalho j.

Morton (1993) define um tipo de linha de produção que designa de *finite-queue flow shop*, caracterizada pela existência de uma capacidade de armazenamento limitada, à entrada de cada processador da linha, excepto na primeira. Além disso, este autor afirma que, um outro caso importante na produção ocorre quando não existem armazéns locais, isto é, quando não é permitida a espera de trabalhos no sistema (ex. linha de produção), com excepção do primeiro processador. Um caso típico ocorre em indústrias de metalomecânica quando, ao processar um determinado metal, se procede ao seu enrolamento, enquanto este ainda se encontra quente. Neste caso um atraso no início do processo de enrolamento do metal pode

ser crítico, uma vez que este processo será dificultado ou impossibilitado se a temperatura baixar abaixo de um determinado valor.

3.3 Ambientes de Escalonamento e de Produção

O conceito de ambiente de produção e ambiente de escalonamento estão relacionados. De uma forma geral e simplificada podemos dizer que o ambiente de escalonamento é determinado pelo ambiente de produção, pela envolvência dinâmica ou estática em que tarefas e sistema são considerados e ainda pela natureza estocástica ou determinística em que as variáveis de escalonamento são consideradas para obtenção de soluções. Desta forma podemos referi-nos a ambientes de escalonamento estáticos e dinâmicos, ambientes estocásticos e determinísticos e ambientes de produção diferenciados pela sua relação com as tarefas. São estas três vertentes classificativas que permitem definir uma diversidade muito alargada de ambientes de escalonamento da produção.

O ambiente de escalonamento é determinante na caracterização de classes de problemas de escalonamento. Cada ambiente define uma classe que por sua vez pode incluir subclasses ou ser um caso particular de outras classes mais abrangentes. As classes de problemas são importantes porque permitem claramente diferenciar a utilidade de métodos ou procedimentos de escalonamento, que geralmente resolvem instâncias de problemas de uma dada classe. É importante perceber que os métodos de escalonamento estão associados aos ambientes de escalonamento onde têm aplicação. Podemos, portanto, organizar uma lista abrangente de métodos em grupos associados a classes de problemas que resolvem, cada uma das quais, por sua vez, associada a cada ambiente de escalonamento identificado.

Convém, desde já, definir método de escalonamento. Se bem que muitas definições poderão ser dadas e diferenças poderão ser identificadas entre o conceito de método e outros conceitos geralmente associados, tais como, algoritmo, procedimento, processo, heurística, meta-heurística e regra, usaremos o termo método para designar qualquer um destes conceitos ou outros similares ou equivalentes usados para obter soluções de escalonamento, quando não se pretende especificamente referir a nenhum em particular mas a qualquer um em geral.

3.3.1 Ambientes de produção

De uma forma simplificada podemos dizer que um ambiente de produção resulta na configuração organizacional do sistema de produção resultante da simbiose entre os

processos de transformação dos trabalhos, expressos nos planos operatórios, e os processadores disponíveis para levarem a cabo essa transformação, eventualmente auxiliados por recursos auxiliares. Portanto, não há ambiente de produção apenas com processadores e meios auxiliares ou apenas com trabalhos e seus respectivos processos, mas sim com a conjugação integrada destas duas entidades, i.e. processadores e trabalhos, tendo em conta as suas características e restrições.

Por os ambientes de produção se referirem ou se poderem identificar com tipos de sistemas de produção, alguns dos quais largamente conhecidos, como o são o *job-shop* e o *flow-shop* (Conway, 1967), usa-se nesta tese de forma equivalente os termos ambiente de produção e de sistema de produção.

Pode-se, desde já, identificar classes muito genéricas de ambientes de produção, nomeadamente:

- uni-fase ou uni-operação e
- multi-fase ou multi-operação;

Estas classes são geralmente aceites por toda a comunidade industrial e académica envolvida no estudo e resolução de problemas de escalonamento.

Os conceitos de fase ou operação, aqui identificados com o mesmo significado, referem-se a uma fase de transformação de um trabalho, que pode ser complexa e se considerar indivisível, podendo envolver um conjunto de operações simples ou elementares, que se realiza num posto de trabalho. Outra operação ou fase necessária no mesmo trabalho obriga à mudança do trabalho do posto em que está. Em alguns casos essa mudança pode dar-se repetindo o uso do posto depois de este sofrer um processo de preparação para iniciar a operação ou fase seguinte de transformação do trabalho, sendo comum, no entanto, operações subsequentes e consecutivas serem realizadas em postos diferenciados.

Definição 3.21 - Ambiente de produção uni-operação (f1): é um ambiente ou sistema de produção caracterizado pelo processamento de trabalhos T_j com uma única operação O_j, também vulgarmente designado de sistema uni-fase. Portanto, o trabalho ou tarefa consiste em processar uma única operação ou fase confundindo-se com ela.

Definição 3.22 - Ambiente de produção multi-operação (fm): é um ambiente ou sistema de produção caracterizado pelo processamento de trabalhos T_j com duas ou mais operações

($O_{ji} \geq= 2$), também vulgarmente designado de sistema multi-fase ou de fases múltiplas. Portanto, os trabalhos integram pelo menos duas fases de produção.

Exemplo de sistema uni-operação ou uni-fase, termos estes que serão usados indistintamente nesta tese (e o mesmo se passando em relação as expressões multi-operação e multi-fase), é um posto de trabalho individual, na área de serviços completamente executando tarefas independentes, em que por exemplo o processador é um computador. De igual modo, pode ser um conjunto de postos de trabalho cada um dos quais capazes de executar quaisquer das tarefas a processar. Por exemplo, uma máquina, ou um conjunto de máquinas injectoras de assentos de plástico, completamente processados apenas numa única operação de injecção, podem também ser considerados sistemas de produção uni-fase.

Vemos, portanto, que a existência de postos de trabalho individuais e autónomos na realização completa de tarefas identificadas como uma única operação é o requisito essencial caracterizador do ambiente de produção uni-operação.

Os sistemas uni-operação podem ser sistemas de processador único ou sistemas de processadores paralelos. Nestes os processadores são equivalentes, i.e. qualquer um pode executar os trabalhos.

A classe dos sistemas multi-operação integra os diferentes tipos ou subclasses de sistemas que são essencialmente distinguidos em função da existência ou não de restrições de precedência entre as operações dos trabalhos que processam, do tipo de relação entre a quantidade de processadores e o número de operações, da existência de processadores repetidos e da existência de processadores flexíveis, como se constatará pelas definições apresentadas a seguir.

Cada um dos sistemas multi-operação descritos abaixo podem, portanto, ser constituídos por processadores flexíveis ou versáteis, dando origem aos correspondentes "sistemas com processadores flexíveis".

Quando existem restrições de precedência arbitrárias entre as operações está-se perante um tipo de ambiente ou sistema de produção multi-operação que é um modelo de produção genérico designado de sistema geral ou *general shop.* Casos particulares deste sistema são a oficina geral (*general job shop*) e a oficina propriamente dita ou oficina pura (*job shop*), de modo análogo, as linhas de produção incluem, tipicamente, o caso da linha geral (*general flow shop*) e da linha pura (*flow shop*) e os sistemas abertos, que também integram os sistemas abertos gerais (*general open shop*) e o sistema aberto puro (*open shop*).

As definições destes sistemas são apresentadas a seguir, que integram as classes e subclasses de sistemas de produção atrás mencionadas.

Definição 3.23 – Sistema de processador único (f1): é um sistema de produção constituído por um único processador, disponível para executar trabalhos com uma única operação (Morton, 1993). Portanto, trata-se de um ambiente uni-operação ou uni-fase. Este é o sistema de produção mais simples ou elementar, em que cada trabalho T_j, $1 \leq j \leq n$, é constituído por uma única operação $O_{ji}=O_{j1}=O_j$, $i=1$, que é realizada em $\mu_{ji}=\mu_{j1}=\mu_j=\{M_i\}=\{M_1\}$.

Num sistema de processador único existe uma única fila de espera de trabalhos, havendo apenas um tipo de decisão a tomar, que consiste na determinação da ordem de execução dos trabalhos, nesse processador (Figura 3.2).

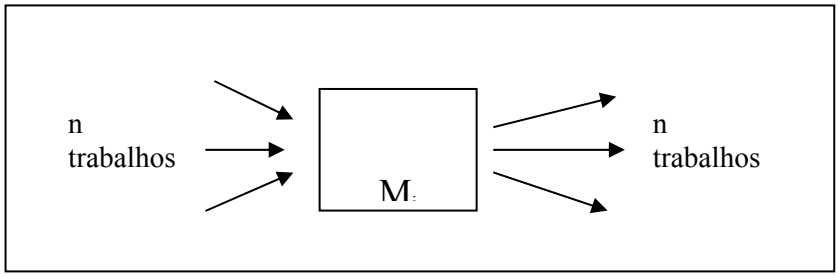

Figura 3.2 – Sistema de processador único.

Definição 3.24 – Sistema de processadores paralelos (f1/P): é um sistema em que cada processador M_i é um elemento do conjunto μ_{ji}, e cada trabalho T_j, $1 \leq j \leq n$, é constituído por uma única operação $O_{ji}=O_{j1}=O_j$, $i=1$, que pode ser processada em um desses processadores $M_i^{(k)} \in \mu_{ji}=\mu_{j1}=\mu_j=\{ M_i^{(1)}, \ldots, M_i^{(k)}\}$, com $k=1, \ldots, m$. Num sistema destes todos os processadores realizam a mesma função (Blazewicz, 1996).

Nestes sistemas além da questão de sequenciação ou ordenação dos trabalhos, anteriormente colocada, para o caso dos sistemas de processador único, também existe o problema da afectação dessas operações aos processadores (Figura 3.3).

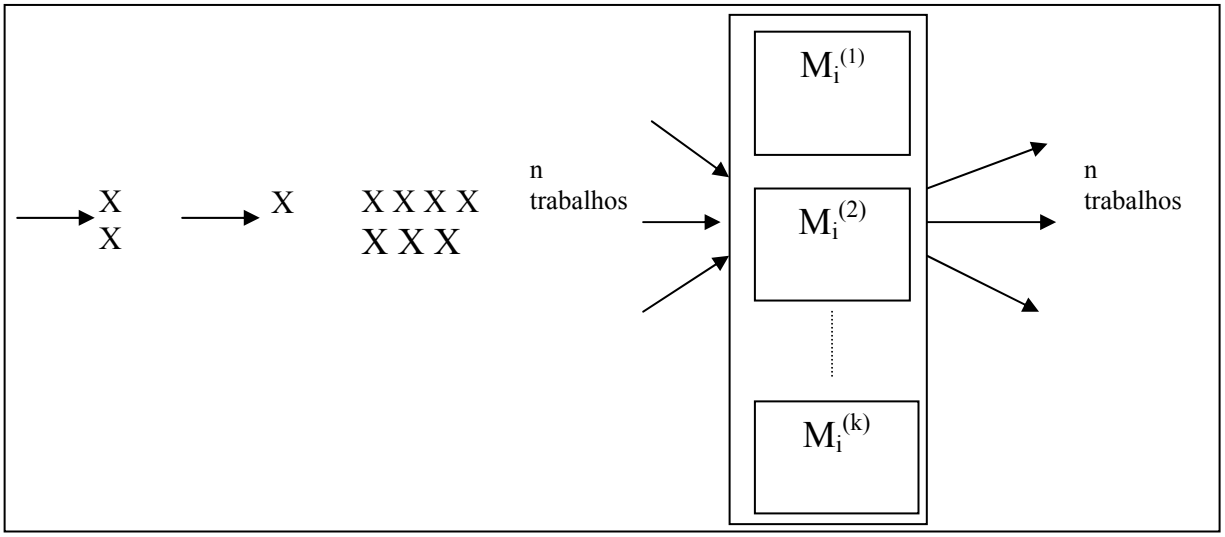

Figura 3.3 – Sistema de processadores paralelos.

Os processadores paralelos podem ainda ser idênticos, uniformes ou não relacionados (Brucker, 1995), isto é, $M_i^{(k)} \in \{PPI, PPU, PPN\}$, conforme se ilustra na Figura 3.4, onde PPI representa processadores paralelos idênticos, PPU representa processadores paralelos uniformes e PPN representa processadores paralelos não relacionados. Dentro da classe de sistemas de processadores paralelos podem, portanto, distinguir-se três subclasses, cujas características variam em função da velocidade e das aptidões de processamento.

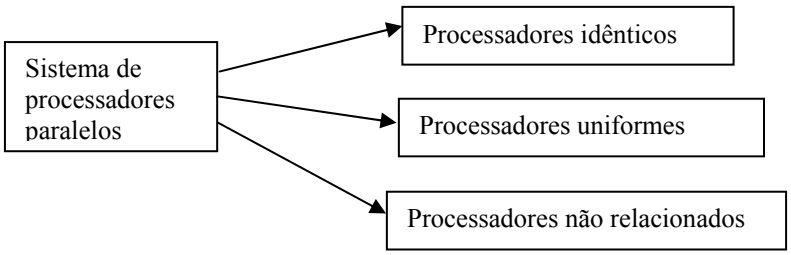

Figura 3.4 - Tipos de processadores paralelos.

Definição 3.25 – Sistema de processadores paralelos idênticos (f1/PI): é um sistema em que os processadores possuem todos a mesma velocidade de execução para cada trabalho/ operação ou, mais precisamente, é um conjunto de processadores que têm o mesmo tempo de processamento, para o trabalho j, dado por, $t_{ji}=t_j$, para todos os processadores $M_i^{(k)}$ (k=1,…,m) do sistema (Brucker, 1995).

Definição 3.26 – Sistema de processadores paralelos uniformes (f1/PU): é um sistema em que os processadores possuem diferentes velocidades de execução para um determinado trabalho/operação, mas cada processador tem uma velocidade de processamento constante

que não depende do trabalho/operação que está a ser realizada, portanto, este tipo de processador não "prefere" qualquer tipo de operação (Blazewicz, 1996). Neste tipo de processadores as velocidades de processamento variam de processador para processador, por um factor de escala v_i: $t_{ji}=t_j/v_i$, para k=1,…,m, onde t_j é o tempo de processamento normalizado (normalmente medido no processador mais lento) e v_i é o factor de velocidade do processador M_i (Brucker, 1995).

Definição 3.27 – Sistema de processadores paralelos não relacionados (f1/PN): é um sistema em que cada processador possui uma velocidade própria de processamento para cada trabalho/ operação. Trata-se de processadores especializados, no sentido em que são mais vocacionados para uns tipos de trabalhos/ operações do que para outro(a)s (Blazewicz, 1996). Portanto, nestes processadores a velocidade de processamento varia não só de processador para processador, como, em cada processador, de trabalho para trabalho ou de operação para operação. Assim, geralmente, $t_{ji}=t_j/ v_{ji}$, para factores de velocidade v_{ji}, de M_i, dependentes do trabalho ou operação a processar nesse processador. Estes processadores poderão, estar providos de um conjunto de instruções, no intuito de permitir a realização de qualquer tipo de trabalho ou operação.

Oficinas

As oficinas são sistemas em que os processadores estão dispostos de tal modo que permitam tirar a máxima vantagem de uma produção mais ou menos diversificada e em quantidades variadas e geralmente em quantidades relativamente pequenas, ou tipicamente mais pequenas do que em linhas de produção. Nestes sistemas os processadores estão, normalmente, dispostos segundo uma lógica funcional, ou seja, agrupados de acordo com as funções que desempenham. Sendo assim, nestes sistemas poderão identificar-se diferentes fluxos de trabalhos.

A oficina é um sistema de produção típico no escalonamento que, contudo, assume diferentes designações por diversos autores. Enquanto, Baker (1974) refere este sistema por *general job shop*, Brucker (1995) designa-o por *general shop*. Outros autores, porém como Blazewicz (1996) e Pinedo (2002) designam-no simplesmente por *job shop*. Dada a divergência nas designações optou-se, neste trabalho, por adoptar a designação de oficina geral, i.e., *general job shop*, para expressar este tipo de ambiente de produção.

Definição 3.28 - Oficina geral (fm/J): é uma oficina sem processadores repetidos, isto é, sem processadores equivalentes alternativos para o processamento das operações.

Neste tipo de sistema é possível voltar a um posto ou processador uma ou mais vezes, durante o processo de produção de um ou vários produtos, como se ilustra na Figura 3.5.

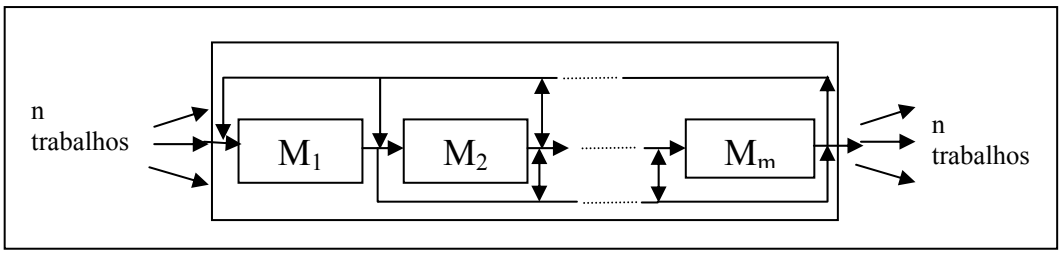

Figura 3.5 – Oficina geral.

Definição 3.29 - Oficina pura[1] **(fm/JP):** é, tipicamente, um sistema em que existe um conjunto de trabalhos T_j a serem processados num número de processadores M_k, cada trabalho consistindo num conjunto de operações O_{ji} a serem processadas numa determinada ordem. Existem, portanto, relações de precedência específicas, do tipo, $O_{j1} \rightarrow O_{j2} \rightarrow O_{j3} \rightarrow \ldots \rightarrow O_{j,nj}$, para qualquer trabalho j = 1, …, n. Assim, a 2ª operação do trabalho T_j só pode ser feita depois da 1ª e assim sucessivamente até à última. Assume-se também que não há processadores repetidos, isto é, $\mu_{ji} \neq \mu_{j,i+1}$, para j = 1, …, n_{j-1}.

O problema das oficinas puras é um caso particular do problema em oficina geral de produção ou *general shop* de Brucker (1995) que, por seu lado generalizam o problema das linhas puras.

Numa oficina existem n trabalhos j=1, …, n e m processadores M_1, …, M_m. Cada trabalho j consiste numa sequência de nj operações, O_{j1}, O_{j2}, …, $O_{j,nj}$, que terão de ser processadas por esta ordem, isto é, existem restrições de precedência na forma $O_{ji} \rightarrow O_{j,i+1}$ (i=1, …, n_j-1). Associada a cada operação O_{ji} existe um processador $\mu_{ji} \in \{M_1, \ldots, M_m\}$ e um tempo de processamento t_{ji}. A operação O_{ji} terá de ser processada, durante t_{ji} unidades de tempo, no processador μ_{ji} . Além disso assume-se que $\mu_{ji} \neq \mu_{j,i+1}$, para j=1, …, n_j-1.

Numa oficina pura, o trabalho j tem n_j operações a serem executadas, ordenadamente, e cada operação está associada a um único processador para a sua realização.

Na oficina pura, tipicamente, um trabalho não visitará mais do que uma vez uma determinado processador além de não existirem grupos de processadores paralelos. Deste

modo, não existirão decisões de roteiro, por não existirem processadores alternativos à execução de uma operação particular, nem alteração da sequência de execução de cada operação.

Uma suposição implícita no problema de oficina é a existência de armazéns locais de capacidade ilimitada. De facto assume-se que um trabalho uma vez tendo terminado o seu processamento num processador poderá esperar fora dele antes de ser processado no processador seguinte. Poderá, contudo, considerar-se a existência de armazéns de capacidade nula o que conduzirá a um problema em que não é permitida a espera dos trabalhos em processamento. Outras restrições podem ser consideradas, quer em termos de trabalhos/ operações quer em termos de processadores, dando origem a problemas diversos, ainda que relacionados.

Linhas

As linhas de produção constituem uma classe importante de ambiente de produção pois trata-se de um tipo de ambiente usual na realidade industrial.

Estes sistemas de produção têm particular relevância e interesse, nomeadamente as ditas linhas puras, dado tratar-se de um tipo de sistema que continua a subsistir e até a manifestar-se, muitas vezes, como um sistema preferencial, dado tratar-se de um sistema de fluxo linear e uni-sentido, com cargas geralmente bem balanceadas, o que lhe confere uma grande produtividade e simplicidade a nível de planeamento e controlo da produção. Neste tipo de sistema o problema que se coloca é o de encontrar uma ordem para os trabalhos em cada processador.

Trata-se, portanto, de um tipo de sistema bem estabelecido na comunidade do escalonamento e geralmente conhecido pela designação de *flow shop* (Baker, 1974, Brucker, 1995, Blazewicz, 1996, French, 1982, Pinedo, 2002, Vollmann, 1997).

Embora neste tipo de ambiente de produção existe, geralmente, a imposição de um fluxo de produção uni-sentido, Morton (1995) generaliza o conceito de linha de modo a incluir os casos de linhas que permitem avançar postos de trabalho (processadores) – *skip flow shop* e o caso de linhas que permitem revisita de um posto de trabalho anterior, voltando ao posto de trabalho imediatamente a seguir àquele em que estava a realizar a última operação – *reentrant flow shop*.

[1] *Classical Job Shop*.

Neste trabalho adoptamos a designação de linha geral para incluir, além da visão clássica de linha, também o caso de avanço de postos de trabalho.

Definição 3.30 – Linha geral (fm/F): é um caso geral de uma linha em que os trabalhos podem avançar processadores e em que, portanto, é possível $n_j \leq m$, para um ou mais trabalhos j (Vollmann, 1997; Morton, 1993). Neste tipo de linha cada conjunto de processadores é correspondente a um determinado estágio ou fase da linha e é constituído por um único elemento, isto é, $\mu_{ji}=\{M_{ji}\}$. Sendo assim, a primeira operação O_{j1}, de cada trabalho j, terá de ser efectuada num processador $M_{j1} \in \{M_1, \ldots, M_{m-nj+1}\}$ e as operações O_{ji} restantes, para i=2, …, n_j, serão processadas em $M_{ji} \in \{M_{k+1}, \ldots, M_{m-nj+i}\}$, em que, k é o processador que executa a operação imediatamente anterior, a cada operação i em causa, em cada instante (Figura 3.6).

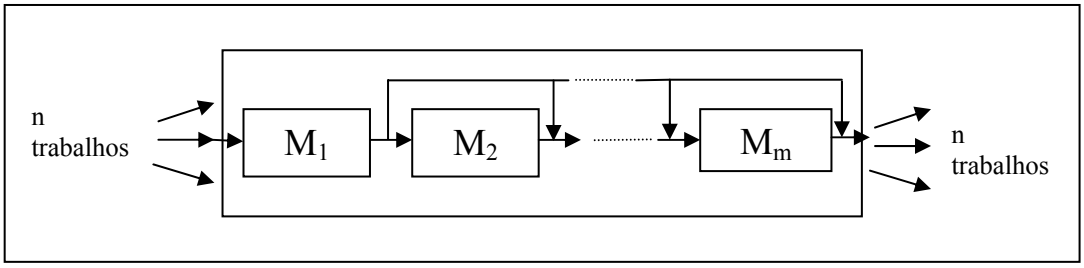

Figura 3.6 – Linha geral.

Definição 3.31 - Linha pura (fm/FP): é um caso particular de uma linha geral, em que existem restrições de precedência entre as operações, na forma $O_{j1} \rightarrow O_{j2} \rightarrow \ldots \rightarrow O_{j,nj}$ e o número de operações é igual ao número de processadores, isto é, $n_j=m$, para cada trabalho j=1, …, n e cada processador $\mu_{ji}=\{M_i\}$, para cada i=1, …, m, e em que, a operação O_{ji} terá de ser processada no processador M_i, qualquer que seja o trabalho j, isto é, existe uma afectação unívoca de cada operação O_i ao correspondente processador M_i, tal que, $(O_i, M_i) \in \{(O_1, M_1), \ldots, (On_j, Mn_j) | 1 \leq i \leq n_j\}$ (Blazewicz, 1996). A Figura 3.7 ilustra o conceito de linha pura.

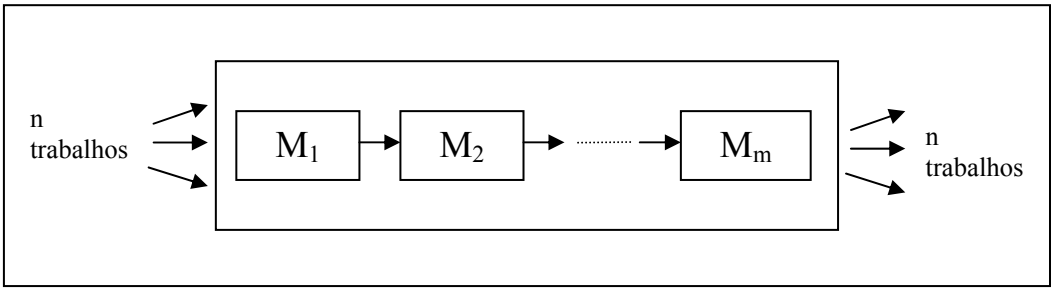

Figura 3.7 – Linha pura.

Um cenário particular que por vezes ocorre em ambiente de linha pura consiste em os trabalhos serem executados por uma mesma ordem, em cada processador, ou seja, a sequência π=(j[1], j[2], …, j[n]), de trabalhos T_j, j=1, …, n, mantém-se invariável para todos os processadores M_i da linha, dando origem a programas de produção ordenados (em cada processador da linha) e tal cenário também é designado, por alguns autores, pela designação de *permutation flow shop* (Brucker, 1995).

Um outro tipo de ambiente referido por alguns autores são os sistemas abertos (Brucker, 1995; Blazewicz, 1996).

Sistemas abertos

Definição 3.32 - Sistema aberto geral (fm/O): é um caso especial de um sistema de produção em que, cada trabalho j consiste num conjunto de operações O_{ji} (i=1,…, k) onde O_{ji} terá de ser processada no processador M_i, sem a imposição de quaisquer restrições de precedência entre as operações (Brucker, 1995).

Definição 3.33 - Sistema aberto puro (fm/OP): é um caso especial de um sistema de produção em que, cada trabalho j consiste em m operações O_{ji} (i=1,…, m) onde O_{ji} terá de ser processada no processador M_i, sem a imposição de quaisquer restrições de precedência entre as operações (Brucker, 1995).

Neste tipo de sistema o problema consiste em encontrar ordens das operações pertencentes a um determinado trabalho e ordens de processadores, ou seja, ordens das operações a serem executadas no mesmo processador.

Uma situação similar verifica-se em oficinas mas, adicionalmente, o processamento de $O_{j,i-1}$ deverá preceder o de O_{ji}, para todos os i=1, 2,…, n_j e para todos os j=1, 2,…, n.

Sendo assim, os sistemas abertos gerais distinguem-se dos puros por estes últimos serem sistemas específicos caracterizados pelo processamento de trabalhos com um número de

operações constante e pelo facto de a primeira operação dever ser realizada num processador, a segunda, num outro processador e assim sucessivamente.

3.3.2 Ambientes de produção flexíveis

Os ambientes de produção anteriormente referidos podem, alternativamente, incluir mais do que um processador em cada posto de trabalho designando-se, neste caso de ambientes ou sistemas de produção flexíveis. Esta característica quando presente num determinado ambiente de produção será expressa por F no início da nomenclatura. Tal característica é referida por Brucker (1995) por *multi-purpose machines* (*mpm*). Este autor considera, portanto, extensões aos sistemas de produção clássicos, quando estes incluem esta característica adicional.

Contudo, nesta designação *mpm*, apresentada por Brucker, confunde-se a existência de múltiplos processadores com a capacidade destes permitirem a execução de operações ou funções diversas. Nós, porém, distinguimos neste trabalho este conceito de existência de múltiplos processadores múltiplos para executar uma operação, que confere a flexibilidade aos ambientes, do conceito de processadores multi-função, que é uma característica adicional e diferente, que pode estar subjacente a qualquer tipo de ambiente de produção, nomeadamente aos ambientes de produção flexíveis.

O conceito de ambiente de produção flexível constitui um aspecto importante num ambiente de escalonamento, na medida em que confere um grau de versatilidade superior a este, uma vez que há mais do que um processador à escolha para executar a operação necessária, independentemente de esta requerer apenas um deles. Esta escolha flexível empresta o nome ao ambiente.

Definição 3.34 – Sistema flexível de processadores paralelos (F/f1/P): é um sistema em que cada trabalho T_j, j=1,…, n, é constituído por uma única operação $O_{ji}=O_{j1}=O_j$, i=1, e pode ser executado em qualquer processador M_i do sistema. A Figura 3.8 ilustra este tipo de ambiente de produção.

Segundo Brucker (1995) cada trabalho pode ser processado por um dos processadores disponíveis em cada conjunto, desde que esteja equipado com as ferramentas apropriadas para a execução desse trabalho.

O processador M_i é um elemento do conjunto $\mu_{js} = \mu_{j1} \cup \mu_{j2} \cup \ldots \cup \mu_{jS}$, $1 \leq s \leq S$, para s subconjuntos de processadores. Cada trabalho T_j, $1 \leq j \leq n$, é constituído por uma única

operação O_j, i=1, e pode ser processada em um desses processadores $M_i^{(s,ks)} \in \mu_{js}$,com, $\mu_{j1} \subseteq \{M_i^{(1,1)}, \ldots, M_i^{(1,k1)}\}, \ldots, \mu_{js} \subseteq \{M_i^{(s,1)}, \ldots, M_i^{(S,ks)}\}$, $k_s=1, \ldots, m_s$. Estes processadores podem ser idênticos, uniformes ou não relacionados, ou seja, $M_i^{(s,ks)} \in PP=\{PPI, PPU, PPN\}$. Deste modo, $PP=\{(\{M_i^{(1,k1)}|1\leq k_1\leq m_1\}, s=1), \ldots,(\{M_i^{(S,ks)}|1\leq k_s\leq m_s\}, s=S)\}=\{\{M_i^{(s,ks)}| 1\leq ks\leq m_s, m_s>1\}|1\leq s\leq S\}$.

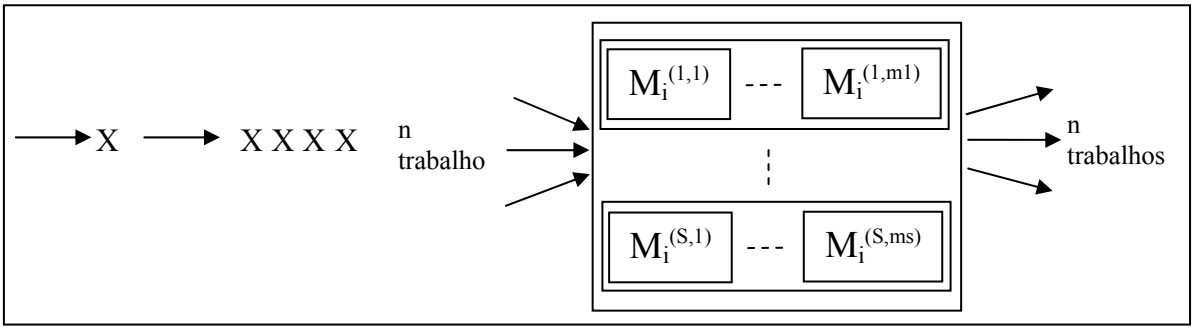

Figura 3.8 – Sistema flexível de processadores paralelos.

Definição 3.35 - Oficina geral flexível (F/fm/J): é uma oficina geral com processadores equivalentes, isto é, com processadores alternativos para o processamento, em que $\mu_{ji}>1$, para pelo menos um trabalho j = 1, …, n e designa-se de oficina geral flexível ou *job shop with machine repetition*, segundo Brucker (1995).

Este ambiente flexível é uma oficina geral em que os diferentes processadores podem ser visitados mais do que uma vez, ou seja, uma oficina em que é possível a existência de fluxos no sentido oposto ao sentido principal do processo produtivo (Morton, 1993). Neste tipo de oficina pode acontecer que $n_j>m$, j=1, …, n. Além disso, quando T_j está num processador $M_k^{(s)} \in \mu_{ji}^{(s)}$, s≥1, de seguida, se não retroceder na oficina, irá para o processador $M_k^{(s+1)} \in \mu_{ji}^{(s+1)}$, efectuar a operação seguinte. Caso contrário, isto é, se T_j estiver actualmente num processador $M_k^{(s),}$ s>1 e, de seguida, retroceder na oficina, com um "salto" ou retrocesso de h≥1, passará para o processador $M_k^{(s')}=M_k^{(s-h)}$, para realizar a próxima operação e, posteriormente, avançará, novamente, para o processador que está na posição $M_k^{(s'+h+1)}$, imediatamente a seguir ao último processador s, em que se encontrava antes de proceder ao retrocesso.

As oficinas gerais flexíveis, na realidade industrial, são sistemas dinâmicos que podem ser vistos como extensões às anteriormente definidas como oficinas gerais e oficinas puras.

Este tipo de sistema revela bastante flexibilidade uma vez que permite voltar a um posto ou processador anterior da oficina, uma ou mais vezes, durante o processo de produção de um

ou vários produtos, dispondo de processadores alternativos para a realização da operação como se descreveu anteriormente e se ilustra na Figura 3.9.

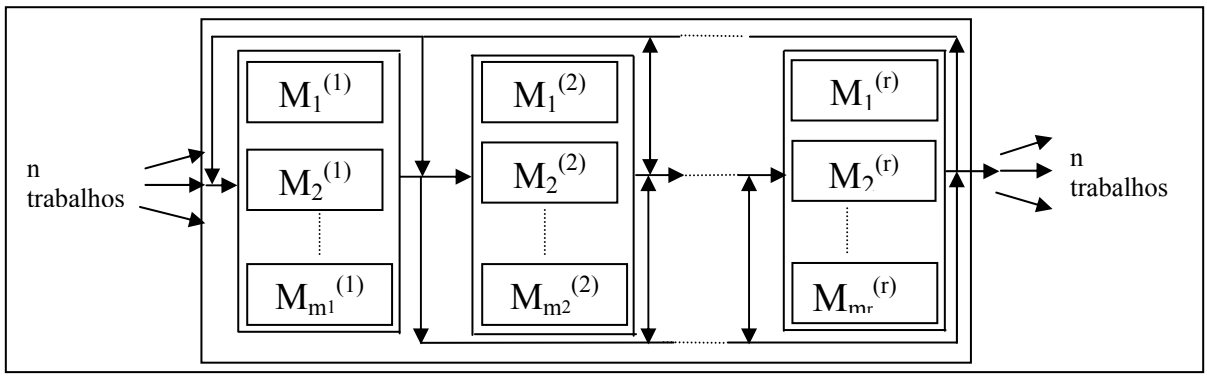

Figura 3.9 – Oficina geral flexível.

Por outro lado, uma generalização da oficina pura permite obter uma oficina pura flexível.

Definição 3.36 - Oficina pura flexível (F/fm/JP): é uma oficina pura com processadores equivalentes, isto é, em cada estágio existe um conjunto de processadores alternativos, que podem ser processadores multi-função ou não, para realizar a operação.

Definição 3.37 – Linha geral flexível (F/fm/F): é uma linha que pode integrar dois ou mais processadores em cada estágio e é geral, podendo haver transposição de estágios (Figura 3.10). Neste tipo de linha cada trabalho T_j, j=1, …, n, é constituído por uma ou mais operações O_{ji}, i=1, …, n_j, e este número de operações é tal que, $n_j \leq r$, em que r é o número de estágios da linha. Além disso, os trabalhos são constituídos por operações em que, $O_{j1} \rightarrow O_{j2} \rightarrow \ldots \rightarrow O_{j,nj}$. Sendo assim, a primeira operação O_{j1} terá de ser efectuada num processador $M_{j1}^{(s)}=M_k^{(s)} \in \mu_{j1}^{(s)}=\{M_k^{(1)},\ldots,M_k^{(r-nj+1)}\}$, para s=1, …, r e as operações O_{ji} restantes, para i=2, …, n_j, são processadas em $M_{ji}^{(s)} \in \mu_{ji}^{(s)}=\{M_k^{(s+1)}, \ldots, M_k^{(r-nj+i)}\}$, em que, s é o estágio que executa a operação imediatamente anterior, a cada operação i em causa.

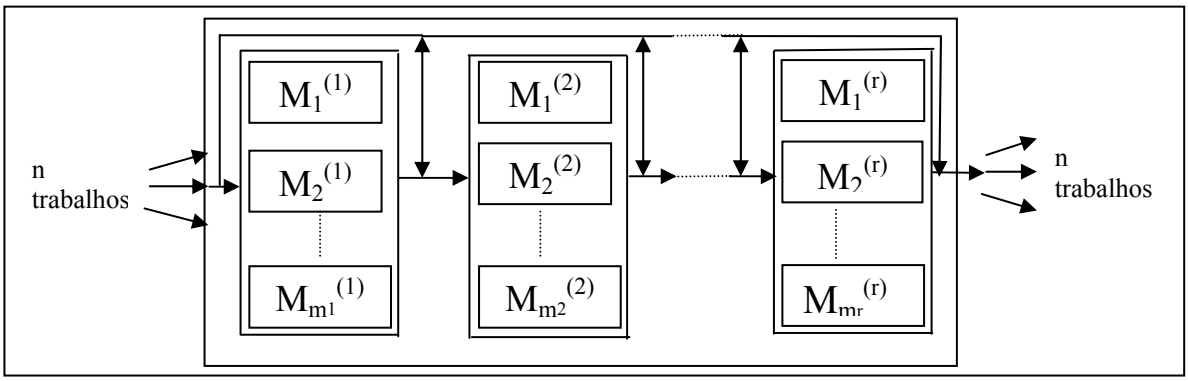

Figura 3.10 – Linha geral flexível.

Definição 3.38 - Linha pura flexível[1] **(F/fm/FP):** é uma linha de em que cada processador ou fase da linha pode ser substituída por um conjunto de processadores paralelos, isto é, equivalentes, sendo uma extensão da linha pura (Figura 3.11) e um caso particular da linha geral flexível. Trata-se de uma linha em que, em cada fase ou estágio do processo produtivo existe um conjunto de processadores capazes de realizar a correspondente operação dessa fase. É uma linha em que n_j=S, para j=1, …, n e O_{ji} é processado num processador $M_i^{(s, ks)} \in \mu_{ji}^{(s)}$, em que $\mu_{ji}^{(s)} \subseteq \{M_i^{(s, 1)}, \ldots, M_i^{(s, ms)}\}$, de cada fase s e s=1, …, S, número de fases da linha, ou seja, $\mu_{ji}^{(s)} \subseteq \{M_i^{(s, ks)}\}$, ks=1,…,$m_s$.

Pinedo (2002) designa este tipo de linha por *flexible flow shop*. Porém, existem várias outras designações, nomeadamente Brucker (1995) que opta pela designação de *multipurpose flow shop* e Morton (1995) que a designa por *compound flow shop*. Outros autores usam ainda outras designações como *flow shop with parallel machines* ou *hibrid flow shop*.

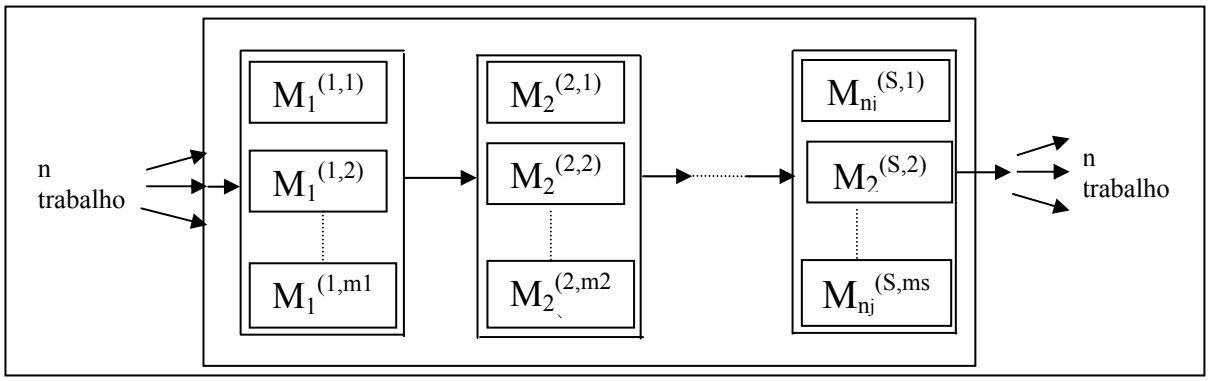

Figura 3.11 - Linha pura flexível.

É de realçar que qualquer sistema de produção pode dar origem ao correspondente sistema multi-função, caracterizado por um determinado grau de multi-fincionalidade, que está dependente da versatilidade dos processadores e recursos de produção e/ ou da sua configuração e funcionamento. Uma possível forma quantitativa de avaliar essa versatilidade será através da variedade de produtos que se espera obter de um determinado sistema desta natureza.

Outra característica importante a considerar nos ambientes de produção é a existência de trabalhos que requerem dois ou mais processadores para a sua execução, como se descreve a seguir.

[1] *Segundo Brucker, Flow-shop with Multi-Purpose Machines (FMPM).*

3.3.3 Ambientes de produção de trabalhos multi-processador

Genericamente podemos dizer que um ambiente de produção de trabalhos multiprocessador (expresso por M, no início da nomenclatura) é um sistema de produção em que há m processadores para a execução de n trabalhos, cada um dos quais com uma ou mais operações sujeitas ou não a relações de precedência, sendo cada operação i de um trabalho j, processada por um ou simultaneamente por dois ou mais processadores de um subconjunto M_{ij} dos m processadores existentes.

Podemos considerar cada um dos sistemas anteriormente referidos com a característica de os trabalhos serem do tipo multi-processador, de onde derivam sistemas como sistemas de processador único, sistemas de processadores paralelos, oficinas e linhas de trabalhos multi-processador.

A existência da característica de trabalhos multi-processador será expressa pela letra M na nomenclatura de representação dos ambientes de produção adoptada neste trabalho. Deste modo, um sistema de processador único, de trabalhos multiprocessador é expresso por M/f1, se for o caso de um sistema de processadores paralelos é representado por M/f1/P. As linhas, oficinas e sistemas abertos puros de trabalhos multi-processador, são representados por, M/fm/FP, M/fm/JP e M/fm/OP, respectivamente. Na secção seguinte descrevem-se os ambientes gerais de produção e apresenta-se uma representação esquemática da nomenclatura subjacente aos ambientes de produção.

3.3.4 Ambientes gerais de produção

Os principais esquemas de classificação de sistemas de produção existentes, em conformidade com o previamente exposto, não identificam claramente alguns tipos de ambientes de produção que surgem na realidade industrial.

A especificação de um modelo que permita a identificação clara e precisa de um vasto conjunto de problemas, que ocorrem em diferentes tipos de ambientes de produção, é um objectivo importante deste trabalho. Houve, portanto, a necessidade de desenvolver uma estrutura classificativa suficientemente detalhada e concisa, por forma a permitir um enquadramento adequado dos problemas de escalonamento no que se refere aos ambientes de produção em que decorrem, dada a inexistência de uma tal classificação na literatura actual.

A estrutura de classificação proposta neste trabalho, ao contrário do que se verifica com a visão dos restantes autores, segue um modelo que parte de um modelo de ambientes de

produção geral flexível de multi-processador (GFM), que vai sendo particularizado à medida que condições específicas, pré-definidas, são especificadas.

Tais condições são relativas a quatro aspectos fundamentais para a identificação de tipos de ambientes. Uma primeira condição é relativa ao número de fases de produção. Deste modo, podem distinguir-se dois casos, que são os sistemas de fase única, expressos pela nomenclatura f1 e os sistemas de fases múltiplas, expressos pela nomenclatura fm. Outro aspecto é relativo à flexibilidade do ambiente de produção, conferida pela existência de processadores alternativos para a execução dos trabalhos, característica esta que é expressa pela letra F na nomenclatura de representação dos ambientes de produção. A característica relativa à existência de trabalhos multi-processador exprime-se pela letra M. Existe ainda a característica relativa à generalidade do sistema de produção, expressa por G.

Um sistema geral (G) é um sistema de produção em que não são impostas restrições de precedência entre operações dos trabalhos sem, contudo, as poder admitir.

Assim, os sistemas de fase única, como são sistemas uni-operação são sistemas não gerais.

De um ambiente de produção geral de fases múltiplas (GFM/fm) podem derivar vários outros ambientes gerais, nomeadamente ambientes gerais flexíveis (GF/fm) e ambientes gerais de trabalhos multi-processador (GM/fm). Casos particulares destes ambientes ou sistemas de produção podem também ser especificados, em função de características relativas ao fluxo de produção e a outras características, como a existência de relações de precedência entre trabalhos/ operações, dando origem a sistemas de produção mais particulares. Se existirem restrições de precedência entre todas as operações num determinado ambiente de produção está-se perante um sistema não geral, nomeadamente sistemas que foram previamente descritos, como as linhas puras (FM/fm/FP, F/fm/FP e M/fm/FP), as oficinas (FM/fm/JP, F/fm/JP e M/fm/JP) e os sistemas abertos puros (FM/fm/OP, F/fm/OP e M/fm/OP).

Na secção seguinte descreve-se o sistema geral flexível de multi-processador (GFM) e as subclasses que derivam directamente deste ambiente geral de produção.

Definição 3.39 - Sistema geral flexível de multi-processador (GFM): é um sistema geral de produção, que é flexível, i.e., em que há *m* processadores para a execução de *n* trabalhos multi-processador, cada um dos quais com uma ou mais operações, com ou sem restrições de precedência sendo, portanto, cada operação *i* de um trabalho *j*, processada por um ou simultaneamente por dois ou mais processadores de um subconjunto M_{ij} dos *m*

processadores existentes. Adicionalmente, os M_{ij} processadores organizam-se num conjunto PM_{ij} de p_{ij} grupos PM_{ij}^{k} (k=1,..., p_{ij}) equivalentes de processadores simultaneamente necessários para processar a operação *i* do trabalho *j*.

Nenhum dos conjuntos PM_{ij}^{k} podem ser usados simultaneamente por mais que um trabalho. A Figura 3.12 ilustra este tipo de ambiente de produção.

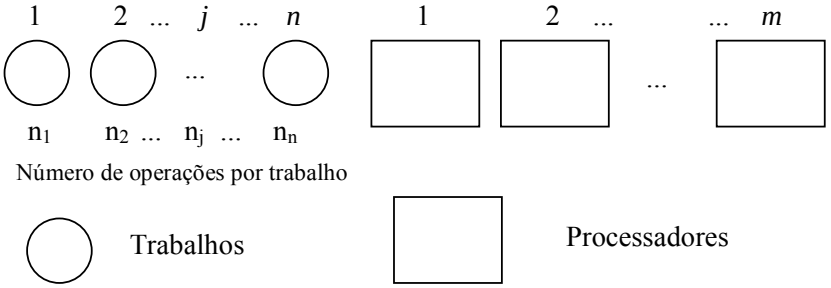

Figura 3.12 – Ilustração GFM.

Portanto, no caso particular de ser necessário apenas um processador para executar a operação, cada grupo PM_{ij}^{k} tem apenas um elemento, i.e. um processador, sendo todos os processadores do conjunto M_{ij} equivalentes e p_{ij} igual ao cardinal do conjunto M_{ij}**.** Neste caso não haverá operações de trabalhos que necessitem do uso simultâneo de dois ou mais processadores, isto é, não existem trabalhos multi-processador. Quando, pelo contrário, a operação *i* do trabalho *j* é executada, simultaneamente, por mais que um processador é necessário identificar o "conjunto PM_{ij} de p_{ij} grupos PM_{ij}^{k} (k=1,..., p_{ij}) equivalentes. Isto deve ser obtido ao equacionar o problema a resolver, i.e. deve ser um dado do problema de escalonamento.

Compreende-se também que a designação de **flexível** para um tal sistema resulta do facto de poder haver processadores alternativos ou equivalentes a usar para a realização das operações e a designação **multi-processador** do facto de se poder ter de utilizar simultaneamente, na realização de pelo menos uma das operações de um trabalho, dois ou mais processadores.

Casos particulares do sistema GFM

Embora existam alguns mecanismos e métodos para resolver problemas de escalonamento pertencentes a este ambiente geral, flexível e de trabalhos multi-processador (GFM) geralmente, restrições impostas neste caso geral configuram uma variedade de casos

particulares frequentemente abordados na literatura, através de procedimentos, métodos e algoritmos específicos.

Esta classe geral GFM pode decompor-se nas classes FM/f1, GFM/fm, GF e GM.

Definição 3.40 - Sistema flexível de multi-processador de fase única (FM/f1): é um caso particular do ambiente GFM sujeito a processamento de trabalhos com apenas uma operação com um conjunto M_j de processadores para cada operação ou trabalho formado do conjunto das m processadores disponíveis.

Adicionalmente, os M_j processadores organizam-se num conjunto PM_j de p_j grupos PM_j^k (k=1,..., p_j) equivalentes de processadores, simultaneamente necessários para processar a única operação do trabalho j. Estes processadores podem resultar de uma combinação de vários tipos, dependendo da natureza da operação (Figura 3.13).

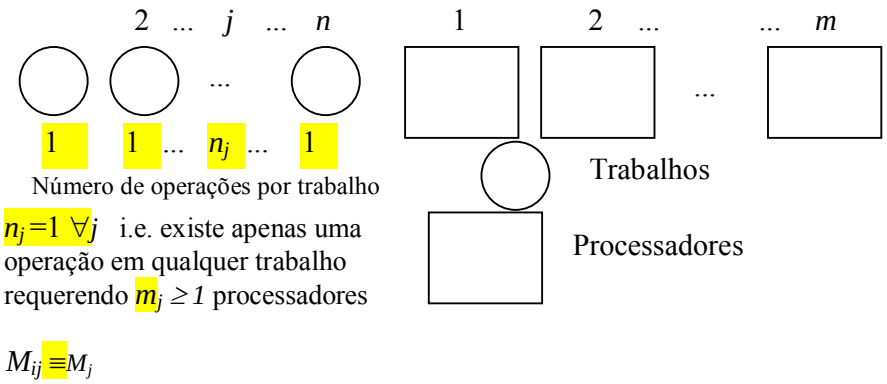

Figura 3.13 – Ilustração FM/f1.

Este sistema admite a existência de processadores equivalentes ou paralelos diferenciados de operação para operação e mesmo a possibilidade de ter grupos de processadores diferentes.

Da classe FM/f1 deriva ainda a classe FM/f1/P, de processadores paralelos.

Definição 3.41 - Sistema flexível de multi-processador de fase única e processadores paralelos (FM/f1/P): é um ambiente FM/f1 sujeito à restrição de os m processadores disponíveis, i.e. o conjunto M serem todos equivalentes, de acordo com as definições de processadores paralelos anteriormente apresentadas.

Assim, os processadores do conjunto M organizam-se num conjunto PM_j de p_j grupos PM_j^k (k=1,..., p_j) equivalentes de processadores simultaneamente necessários para

processar a operação do trabalho *j*. Neste caso de sistema de processadores paralelos, os processadores m_j são também equivalentes. Isto significa que: $M_j = M = \{1,2,...,m\} \quad \forall_j$ com $m_j \geq 1$.

Casos particulares do sistema FM/f1/P

A classe FM/f1/P divide-se nas classes FM/f1/PI, FM/f1/PU e FM/f1/PN, de processadores paralelos idênticos, independentes e não relacionados, respectivamente.

Definição 3.42 - Sistema flexível de multi-processador de fase única e processadores paralelos idênticos (FM/f1/PI): é uma instância do ambiente FM/f1/P sujeito a processamento de trabalhos no conjunto M dos m processadores disponíveis todos equivalentes idênticos, sendo por isso o tempo de processamento t_j de cada trabalho j, igual em qualquer dos grupos PM_j^k (k=1,..., pj) equivalentes idênticos de processadores, i.e. $t_{jk}=t_j$ $\forall k=1,..., p_j, ; j=1,...,n$.

Definição 3.43 - Sistema flexível de multi-processador de fase única e processadores paralelos uniformes (FM/f1/PU): é um sistema em que o ambiente MF/f1/P é instanciado para processamento de trabalhos no conjunto *M* dos *m* processadores disponíveis todos equivalentes uniformes, sendo, por isso, o tempo de processamento t_{jk} de um trabalho *j* por um grupo PM_j^k (k=1,..., p_j) de m_j processadores proporcional ao tempo de execução do trabalho, i.e. da sua única operação, por outro grupo. Isto equivale a dizer que, havendo um grupo de processadores usados para simultaneamente executar o trabalho *j* no tempo t_j este tempo é superior v_k vezes ao tempo de execução do mesmo trabalho no grupo PM_j^k. A v_k podemos chamar factor de rapidez de execução do grupo *k*, i.e, PM_j^k. Geralmente, este factor faz-se igual a 1 no grupo de processadores mais lento que processa a operação no tempo t_j .

Portanto: $t_{jk} = t_j / v_k$

em que t_j é o tempo de execução do trabalho *j* no grupo de processadores mais lento

e v_k é o factor de rapidez de execução do grupo *k*. Como se compreende este factor é independente do trabalho *j*.

Definição 3.44 - Sistema flexível de multi-processador de fase única e processadores paralelos não relacionados (FM/f1/PN): trata-se de um sistema em que o ambiente

FM/f1/P é sujeito a processamento de trabalhos pelo conjunto *M* de *m* processadores disponíveis mas todos equivalentes não relacionados. Por isso, o tempo t_{jk} de processamento de um trabalho, ou da sua única operação, não está proporcionalmente relacionado com o grupo PM_j^k de processadores. Podemos, no entanto, exprimir o tempo t_{jk}, de execução de um trabalho *j* num grupo PM_j^k de processadores, em função do tempo de execução do mesmo trabalho no grupo de processadores onde demora mais tempo a executar, i.e. demora t_j, com base no factor de rapidez v_{jk} de execução do trabalho *j* no grupo PM_j^k. Estamos a assumir que esta rapidez de execução de um trabalho, que nos grupos de processadores uniformes era apenas dependente do grupo, é agora também dependente do trabalho em questão. Isto significa que o grupo equivalente de processadores mais lento usados para simultaneamente executar o trabalho *j* no tempo t_j não é necessariamente o mais lento a executar um outro trabalho. Portanto: $t_{jk} = t_j / v_{jk}$:

sendo t_j o maior tempo de execução do trabalho e

$\boldsymbol{v_{jk}}$ o factor de rapidez de execução do trabalho *j* no grupo PM_j^k .

Como se compreende este factor $\boldsymbol{v_{jk}}$ é não só dependente do trabalho, i.e. da sua operação, mas também do grupo de processadores que em simultâneo o executam. Esta dependência do trabalho *j* não existe em grupos equivalentes de processadores uniformes.

Do ambiente de produção general flexível de trabalhos multi-processador, GFM podem, portanto, derivar um conjunto de ambientes de produção, GFM/fm, GFM/fm/J, FM/fm/JP, GFM/fm/F, FM/fm/FP, GFM/fm/O e FM/fm/OP, que representam sistemas de produção que se podem derivar de sistemas de múltiplas fases. Tais sistemas são derivações dos sistemas clássicos anteriormente apresentados, como as oficinas (*job-shops*), as linhas (*flow-shops*) e os sistemas abertos (*open-shops*).

Desta classe geral GFM deriva também a classe geral GM que, por seu lado, se decompõe nas classes M/f1 e GM/fm. A classe M/f1 origina ainda a classe geral M/f1/P (note-se que também é uma classe geral porque não está especificado o tipo de processadores paralelos, P), de onde ainda derivam as classes M/f1/PI, M/f1/PU e M/f1/PN. A classe GM/fm, à semelhança da classe GFM, dá origem às subclasses GM/fm/J, M/fm/JP, GM/fm/F, M/fm/FP, GM/fm/O e M/fm/OP.

De igual modo, da classe geral GFM deriva a classe GF e seus ambientes relacionados, F/f1 e GF/fm. A classe F/f1 dá origem à classe geral F/f1/P, de onde ainda derivam as classes

F/f1/PI, F/f1/PU e F/f1/PN. A classe GF/fm ainda origina as subclasses GF/fm/J, F/fm/JP, GF/fm/F, F/fm/FP, GF/fm/O e F/fm/OP.

Os casos GM e GF seguem, portanto, uma estrutura similar. Enquanto GM são sistemas gerais com trabalhos do tipo multi-processador, os sistemas GF são caracterizados pela inclusão de postos de trabalho flexíveis de multi-processador.

Além disso, se um sistema deixa de ser flexível, por não haver escolha alternativa do processador para a realização da operação, a letra F é removida da nomenclatura. De igual modo se adicionalmente se deixar de verificar a existência de trabalhos multi-processador, então também é removida a letra M da nomenclatura e o ambiente de produção resultante é simplesmente representado pela letra G, seguida da nomenclatura que especifica o tipo de ambiente multi-operação subjacente. Alternativamente, pode tratar-se de um ambiente uni-operação, não geral, nomeadamente, sistemas de processador único ou de processadores paralelos idênticos, uniformes ou não relacionados. A Figura 3.14 sintetiza a estrutura hierárquica global da nomenclatura de ambientes de produção propostos.

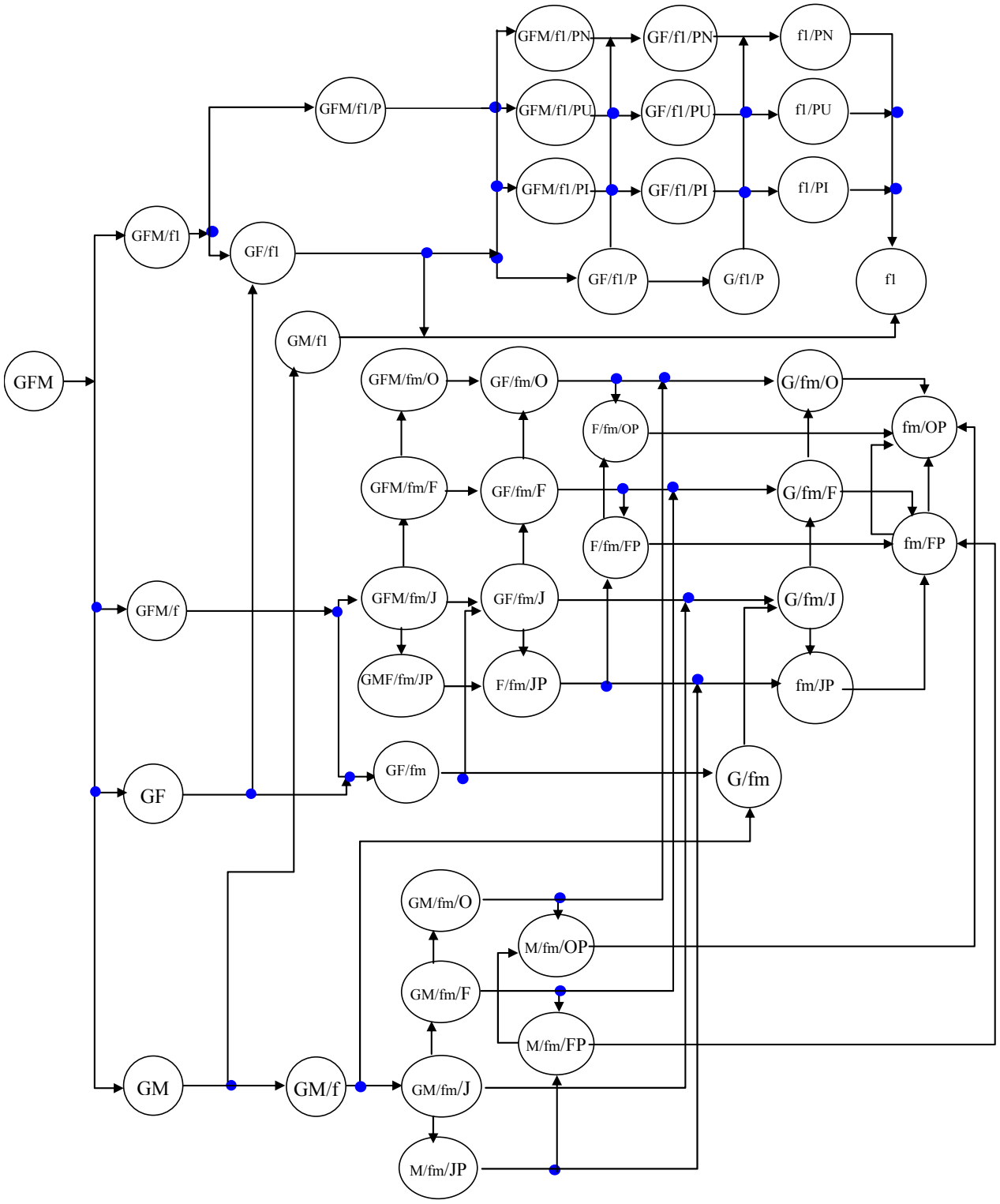

Figura 3.14 –Ambientes de produção e sua inter-relação

LEGENDA:

G Geral

F Flexível, i.e. **que pode ter** um ou mais processadores equivalentes

M Multi-processador, i.e. trabalhos **podem ter** operações a processar com um ou mais processadores em simultâneo.

fl Fase única. Cada trabalho apenas **tem uma operação**, i.e. $n_j=1$ $\forall j: j=1, ..., n$

fm Múltiplas fases. Cada trabalho **tem duas ou mais operações**, $n_j \geq 2$ $\forall j : i=1, ..., n$

P Processadores equivalentes, i.e paralelos. O sistema **tem dois ou mais processadores** equivalentes

PN Processadores equivalentes **não relacionados**

PU Processadores equivalentes **uniformes**

PI Processadores equivalentes **idênticos**

J: Jobshop; JP: Pure jobshop; F: Flowshop; FP: Pure flowshop; O: Openshop; OP: Pure openshop.

Figura 3.14 –Ambientes de produção e sua inter-relação (continuação).

3.4 Critérios de optimização

Os critérios de optimização são utilizados no escalonamento como um dos principais parâmetros para a selecção de um método adequado à resolução de um problema.

Um bom programa de produção é aquele que permite satisfazer um objectivo, associado ao desempenho do sistema de produção. Este desempenho é, usualmente, avaliado através de critérios de optimização que, tipicamente, consideram a utilização do sistema de produção, o fluxo de produção, o atraso dos trabalhos ou algum critério económico.

Os critérios de optimização são critérios baseados em medidas para avaliar o desempenho ou a eficiência de um sistema de produção, isto é, medidas necessárias para avaliar a qualidade das decisões e do funcionamento de um sistema de produção, sendo também designados por critérios de optimização ou de eficiência. Em causa está a minimização ou a maximização desses critérios. Estes podem ser simples ou combinados ou compostos e são expressos através de uma função objectivo mais ou menos elaborada. Daqui em diante usar-se-á a expressão "critérios de optimização" para referir, genericamente, essas medidas.

Os critérios de optimização permitem atingir diversos tipos de objectivos, nomeadamente:

- Maximizar o fluxo produtivo num determinado período.
- Satisfazer os requisitos de qualidade e a rapidez de resposta aos clientes (como o cumprimento de prazos de entrega de trabalhos).
- Minimizar custos de produção.

- Combinações dos casos anteriores, entre muitos outros.

Definição 3.35 - Critério de optimização (CO): pode ser definido através de uma função objectivo do tipo Max/ min $\sum ([w_j] * Z_j)$, em que, w_j é o parâmetro de prioridade atribuído ao trabalho j, podendo ou não existir, e Z_j é o critério específico em causa (Baker, 1974).

3.4.1 Classificação de critérios de optimização

Uma forma de classificar os COs pode ser quanto à sua regularidade e à sua complexidade. Um CO pode definir-se como regular, não regular, simples ou combinado (Figura 3.15).

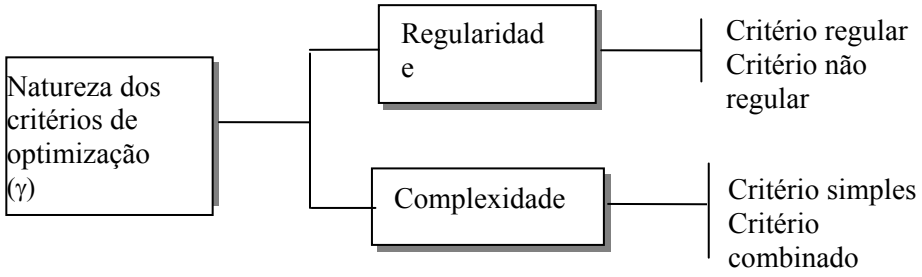

Figura 3.15 - Natureza dos critérios de optimização.

Os critérios de optimização regulares podem ainda ser subdivididos nas seguintes categorias, como se sintetiza na Figura 3.16 (Morton, 1993):

- Utilização do sistema
- Fluxo produtivo
- Atraso
- Critério económico

Definição 3.36 - Critério [de optimização] regular (CR): é um critério Z em que, o objectivo consiste em minimizá-lo e Z pode crescer somente quando pelo menos um dos tempos de conclusão de processamento Cj, de um trabalho j cresce, isto é: se $Z=f(C_1, C_2, \ldots, C_n)$, para uma sequência S e $Z'=f(C'_1, C'_2, \ldots, C'_n)$ para uma sequência S', então Z é regular se: $Z'>Z \Rightarrow C'_j>C_j$, para algum trabalho j (Baker, 1974).

Na optimização de qualquer CR é suficiente considerar os programas activos (Baker, 1974).

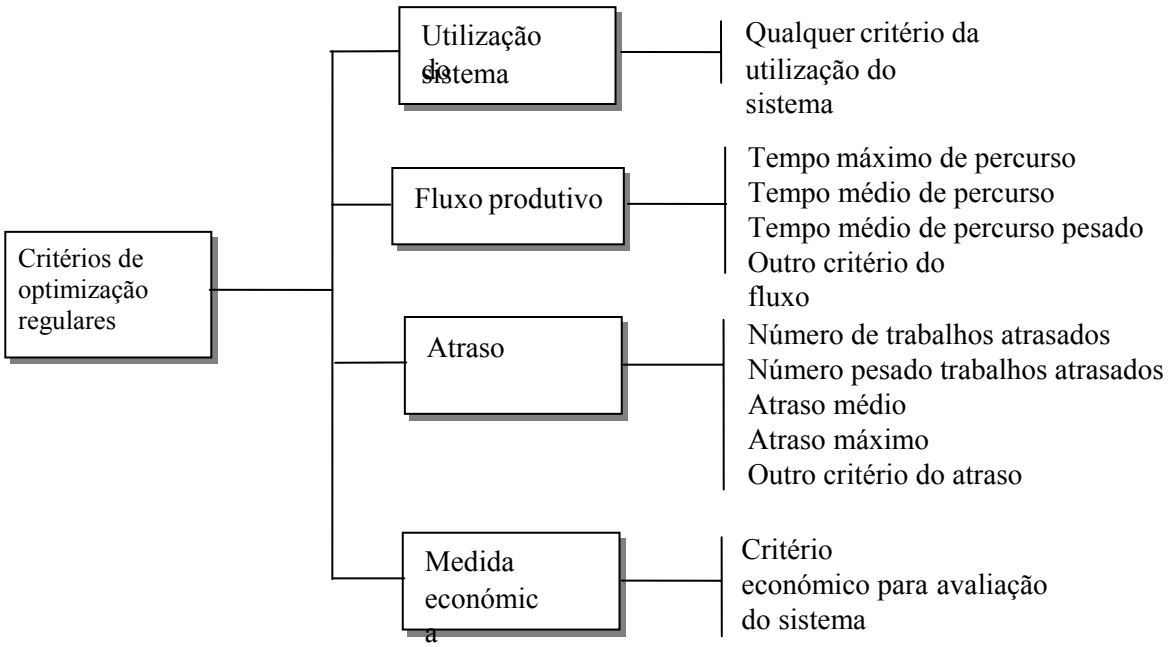

Figura 3.16 - Tipos de critérios de optimização regulares.

Por outras palavras poderá dizer-se que um critério é considerado regular quando implica preferir sempre terminar um trabalho ou operação o mais cedo possível, em vez de considerar a hipótese de o/ a atrasar e, normalmente, o objectivo consiste em minimizá-lo.

Podemos considerar as seguintes situações (Morton, 1993):

a) maximizar algum critério de utilização do sistema;

b) minimizar algum critério do tempo de percurso de fabrico dos trabalhos no sistema;

c) minimizar algum critério do atraso dos trabalhos;

d) minimizar algum critério dos custos do sistema.

Sendo assim, serão exemplos de critérios de optimização do primeiro caso:

a) qualquer critério da utilização do sistema;

b) o instante máximo de conclusão dos trabalhos em curso de fabrico ou *makespan*, o tempo de percurso pesado ou *weighted flowtime* e o tempo máximo de percurso ou em curso de produção ou *maximum flowtime*; c) o número pesado, ou não, de trabalhos ou operações atrasadas ou [*weighted*] *number of tardy jobs*, o *weighted lateness* / *tardiness*, *lateness*, *tardiness* e outros critérios relacionados com o atraso, pesado ou não.

d) qualquer critério económico do sistema.

Quando estamos perante critérios não regulares pode não ser assim, por exemplo, num ambiente de produção do tipo *JIT*[1], terminar os trabalhos demasiadamente cedo poderá representar excesso de trabalho em curso ou *WIP*[2], o que será altamente desvantajoso! Por outro lado o cliente pode não estar interessado em receber a sua encomenda antes do prazo previsto!

Um exemplo de um critério de optimização não regular apropriado é o critério *weighted early-tardy objective*. Muito importante quando os clientes não querem trabalhos atrasados mas também não os querem receber "cedo de mais" (portanto, segundo um dos princípios da filosofia *JIT*).

3.4.2 Critérios de optimização e relações típicas

Um programa de produção no qual o valor de um critério de optimização particular γ assume o seu valor mínimo será designado óptimo e o correspondente valor de γ será representado por γ^*.

As seguintes medidas podem ser calculadas para cada trabalho T_j, j=1, 2, …, n, considerados num determinado programa.

- Tempo ou instante de conclusão ou *completion time*, C_j, do trabalho j;
- Tempo de percurso ou em curso de produção ou *flow time*, $F_j = C_j - r_j$, sendo a soma de tempos de espera e de processamento;
- Tempo de percurso é o tempo de permanência ou em curso dos trabalhos no sistema de produção. O tempo de percurso poderá também ser designado de tempo de fabrico ou *lead time* de fabrico.
- *Lateness*, $L_j = C_j - d_j$
- Atraso ou *tardiness*, $T_j = \max \{C_j - d_j, 0\}$;
- Adianto ou *earliness*, $E_j = \max \{d_j - C_j, 0\}$;

em que d_j representa a data de entrega devida do trabalho j.

3.4.2.1 Exemplos de critérios

Para avaliar os programas de produção usam-se frequentemente quatro tipos fundamentais de critérios de optimização:

[1] *Just-in-time*.
[2] *Work in process.*

O tempo de percurso ou em curso máximo, que é o tempo total de produção de um dado número de trabalhos (n) (ou de lotes n_l) e corresponde ao período de tempo, desde o início do processamento do primeiro até à conclusão do último (por exemplo, um mês), isto é, indica o tempo de percurso total ou máximo de um conjunto n de trabalhos T_j, F_{max}, ao fim do qual todos estão concluídos.

Quando o interesse do critério de optimização do sistema de produção recai no instante máximo ou final, ao fim do qual o conjunto de trabalhos considerado está concluído, então em vez de F_{max} usar-se-á o correspondente critério C_{max}, que expressa o comprimento do programa, isto é, o instante máximo de conclusão dos trabalhos no sistema ou *makespan*, $C_{max}= \max\{C_j\}$, em que C_j é o instante de conclusão de cada trabalho j ($1 \leq j \leq n$) e C_{max} é o instante de conclusão do último trabalho processado, quando todos os trabalhos são lançados no sistema, a partir de um determinado instante de chegada ao sistema r_j ($1 \leq j \leq n$), a partir do qual cada trabalho fica disponível para ser processado.

Em ambiente estático de produção, isto é, quando $r_j=0$, para todos os trabalhos T_j ($1 \leq j \leq n$), $C_{max}= F_{max}$, dado que, $C_j= r_j + F_j$.

O problema da minimização do C_{max} ou *makespan* é um problema de escalonamento importante. A resolução deste problema traduz-se na rápida execução dos trabalhos e também numa boa utilização do sistema de produção, isto é, dos processadores deste, sendo um critério de optimização típico, muito utilizado no escalonamento.

Os valores médios, F_{med} e C_{med}, são também critérios de optimização muito ususais e correspondem, respectivamente ao tempo de percurso médio ou *mean flow time*, $F_{med} = 1/n \sum_{j=1..n} F_j$, em que se pretende minimizar o tempo de fabrico médio por trabalho e ao $C_{med} = 1/n \sum_{j=1..n} C_j$.

Critérios típicos expressos em termos de tempo total de processamento dos trabalhos, pesado ou não por um factor w_j têm a as seguintes expressões:

- $\sum(j=1 \text{ a } n)F_j$;
- $\sum(j=1 \text{ a } n)C_j$;
- $\sum(j=1 \text{ a } n)w_jF_j$;
- $\sum(j=1 \text{ a } n)w_jC_j$.

Outros critérios são relacionados com medidas que consideram datas ou prazos de entrega ou de conclusão, dj, dos trabalhos Tj, onde se destacam os mais típicos, como o atraso

máximo ou maximum lateness, Lmax=max{Lj} e o correspondente atraso pesado dos trabalhos Lwmed = ∑j=1..n wjLj / ∑j=1..n wj, onde, Lj:=Cj-dj (atraso); para 1≤j≤n.

Critérios de optimização usuais com prazos associados aos trabalhos são:

- E_j=max {0, d_i-C_i};
- T_j=max {0, C_i-d_i};
- D_j=|C_j-d_j| (desvio absoluto);
- S_i=(C_i-d_i)^2 (desvio quadrado);
- U_j=0, se C_j≤d_j e U_j=1, no caso contrário (penalização unitária).

Em algumas aplicações poderão ser utilizados outros critérios, por exemplo:

- Atraso positivo médio ou *mean tardiness*, T_{med}= 1/n $\sum_{j=1..n} T_j$,
- Atraso positivo médio pesado ou *mean weighted tardiness*, $T_{wmed} = \sum_{j=1..n} w_j T_j / \sum_{j=1..n} w_j$;
- Atraso negativo/adianto médio ou *mean earliness*, $E_{med} = 1/n \sum_{j=1..n} E_j$;
- Atraso negativo/adianto médio pesado ou *mean weighted earliness*, $E_{wmed} = \sum_{j=1..n} w_j E_j / \sum_{j=1..n} wj$;
- Número de trabalhos ou lotes atrasados ou *tardy jobs*, $N_T = \sum_{j=1..n} \partial(Lj)$, onde $\partial(x) = 1$ se $x > 0$, e $\partial(x) = 0$ se $x \leq 0$;
- Número pesado de trabalhos atrasados $N_{Twmed} = \sum_{j=1..n} w_j N_{Tj}$.
- Tempo médio de percurso pesado ou *weighted mean flow time*, $F_{wmed}=\sum_{j=1..n} w_j F_j / \sum_{j=1..n} w_j$

Os anteriores valores de w_j, correspondem a um valor de "peso", que pondera a importância ou urgência de cada trabalho j e, deste modo, estes poderão ser distinguidos, em cada tipo de critério de optimização a que correspondam, do seguinte modo:

- *weighted flowtime*: w_j;
- *weighted tardiness*: $w_j T_j$;
- *weighted earliness*: $w_j E_j$;
- *weighted number of tardy jobs*: $w_j N_{Tj}$

De entre os critérios de optimização mais usuais destaca-se a minimização do "comprimento do programa" (*Schedule Length - Cmax*), que corresponde à minimização do instante máximo de conclusão dos trabalhos no sistema.

O critério do tempo médio de percurso é também importante, do ponto de vista dos utilizadores, uma vez que a sua minimização permite obter melhorias a nível da minimização do tempo médio de resposta do sistema às solicitações dos clientes, através da maximização do fluxo ou cadência produtiva do sistema.

Os critérios que envolvem prazos de entrega continuam a ser de primordial importância em sistemas de produção, especialmente nos que produzem para encomenda.

A maior parte das métodos que têm sido desenvolvidos até hoje são referentes ao cumprimento de prazos ou datas de entrega e à minimização do tempo máximo de percurso ou de em curso de fabrico dos trabalhos no sistema de produção, embora existam métodos susceptíveis de utilizar qualquer um dos critérios referidos anteriormente, ou outros critérios não mencionados, quer de uma forma simples ou combinada.

3.4.2.2 Relações fundamentais entre critérios

Num problema estático, ou seja, em que se considera que todos os trabalhos ficam simultaneamente disponíveis para serem processados, a partir de um determinado instante (normalmente num instante 0), isto é, um problema em que se verifica a condição $chega_j=0$, $\forall_{j\in T}$, podem estabelecer-se as seguintes relações matemáticas entre critérios de optimização:

$F_{med} * n = J_{med} * F_{max}$, com, J_{med} o número médio de trabalhos em curso, no tempo total de percurso, dos n trabalhos, que é igual a F_{max} e F_{med} é o tempo médio de percurso por trabalho.

$F_{wmed} * \sum_{j=1..n} w_j = V_{med} * F_{max}$, com, V_{med} o valor (monetário, por exemplo) médio dos trabalhos em curso durante o tempo total de percurso F_{max}.

Esta relação não é mais do que uma generalização da relação anterior, em que um factor w_j "pesa" cada trabalho j (Baker, 1974).

3.5 Diversidade de problemas de escalonamento

Os problemas de escalonamento são problemas muito diversos, existindo imensas formas de os resolver, de entre as quais se pretende, geralmente, obter um solução que seja óptima. Sendo assim, podem identificar-se diferentes tipos de problemas, como se descreve a seguir.

3.5.1 Problema clássico

Existem duas restrições gerais na teoria do escalonamento clássico. Cada operação é executada por um único processador de cada vez, eventualmente com o uso de recursos auxiliares, e cada processador é capaz de processar no máximo uma operação de cada vez.

Os problemas clássicos estão geralmente relacionados com restrições de precedência e/ou de processadores e, mesmo sem considerar tarefas preparatórias do sistema ou dos processadores são, frequentemente, problemas bastante complexos.

Pode dizer-se que, um problema de escalonamento "clássico" (PEC) pode ser descrito pela existência de trabalhos com operações pré-definidas e processadores, também pré-definidos, com dados temporais relativos, nomeadamente, à chegada de cada trabalho ao sistema e à disponibilidade de cada processador e um critério de optimização ou objectivo, envolvendo o tempo de conclusão dos trabalhos. Este foi o problema mais abordado no passado.

3.5.2 Problema expandido

Se se adicionarem outras características ao problema clássico, poderá falar-se do problema de escalonamento expandido (PEE). Uma dessas considerações prende-se com as estratégias de planeamento de capacidade e consiste em mudar a quantidade e/ou configuração interna ou externa dos meios de produção à medida que o problema progride, no sentido de compensar as alterações de carga do sistema. De um modo similar poder-se-ão reconfigurar os trabalhos e as operações. Por exemplo, poderá haver várias alternativas relativamente às tecnologias a utilizar para realizar uma dada operação, requerendo diferentes quantidades de processadores e mais ou menos tempo de execução como resultado, sendo assim, aspectos relacionados com o planeamento de processos de produção podem também ser equacionados.

Uma reconfiguração externa consistirá, por exemplo, em negociar, com os clientes, prazos ou datas de entrega e custos de penalização, com base na projecção de capacidade da fábrica e concorrência, entre outros factores.

Adicionalmente, segundo Jordan (1996), poderá distinguir-se entre programação com e sem formação de famílias de produtos e/ou de agrupamento de trabalhos em lotes, ou de programação de trabalhos independentes e individuais, isto é, de lotes unitários de trabalhos.

A maioria da literatura sobre escalonamento da remete, porém, para a programação de produtos independentes e individuais. Para a programação com formação de famílias e lotes de trabalhos começa, porém, também, cada vez mais, a existir literatura disponível, com referência a métodos adequados para resolver estes tipos de problemas (Jordan, 1996). Em primeiro lugar, se o agrupamento de trabalhos remete para assuntos relacionados com a dependência da sequência ou ordem de lançamento dos trabalhos (em termos de preparação dos processadores), então os problemas são, frequentemente, abordados no contexto do problema do caixeiro viajante (PCV), em segundo lugar, formar grupos de trabalhos é frequentemente identificado como sendo um problema de dimensionamento de lotes, e nesta área existe já muita investigação desenvolvida. Em terceiro lugar não existe um problema normalizado, tal como o problema das oficinas ou o PCV. Consequentemente, a terminologia não é homogénea e torna-se difícil comparar diferentes abordagens alternativas.

Outras questões que podem ser consideradas num problema expandido (PEE) prendem-se, por exemplo, com a problemática de balanceamento de linhas de produção (Jordan, 1996).

3.5.3 Problema estático versus dinâmico

O cenário mais frequente que ocorre na realidade industrial é dinâmico e daí os problemas de escalonamento serem geralmente dinâmicos (PED). Estes problemas podem ser definidos, de um modo muito simples e geral, como problemas em que os dados se alteram no tempo.

Em tais ambientes reais, modificações mais ou menos imprevistas e aleatórias podem ocorrer continuamente, conduzindo a alterações nos problemas. Sendo assim, trabalhos mais ou menos complexos têm de ser processados em sistemas de produção mais ou menos complexos, sob ambientes dinâmicos, isto é, frequentemente sujeitos a diversos tipos de ocorrências e perturbações aleatórias, tais como, chegadas de novos trabalhos, avarias de processadores, absentismo de operários, cancelamento de trabalhos e alterações de prazos de entrega e/ ou dos tempos de processamento dos trabalhos, entre outros.

Numa visão mais geral, as alterações dinâmicas num problema podem ser vistas como a adição e/ ou a remoção ou a actualização de restrições. Mais especificamente, nestes problemas de escalonamento, isto pode ser expresso através da adição ou remoção de um conjunto de trabalhos ou a restrição ou relaxamento da janela de tempo dos trabalhos a processar.

3.5.4 Problema determinístico versus não determinístico

Os problemas de escalonamento podem ainda ser tratados como determinísticos ou não determinísticos. Esta forma de designação ou categorização destes problemas prende-se com a natureza dos dados. Os problemas determinísticos são problemas em que os dados são determinísticos, isto é, têm um valor bem determinado e não variam, nomeadamente, os instantes de chegada dos trabalhos ao sistema e todos os restantes dados, que são conhecidos e fixos. Pelo contrário, há casos em que as variáveis do problema não são bem definidos, podendo haver incerteza na sua determinação e, desta forma, poder-se-á estar perante problemas de natureza estocástica ou de natureza difusa. Um exemplo típico de problemas não-determinísticos são, por exemplo, os problemas em que os instantes de chegada dos trabalhos ao sistema são aleatórios e/ou os tempos de processamento dos trabalhos são expressos por distribuições estocásticas.

Nas secções que se seguem apresenta-se uma análise geral à complexidade dos problemas de escalonamento e formas de contornar a complexidade da sua resolução.

3.6 Análise da complexidade dos problemas de escalonamento

A diversidade dos problemas de escalonamento, as suas dimensões de larga escala e a sua natureza dinâmica tornam estes problemas muito complexos e computacionalmente difíceis de resolver. Os problemas de escalonamento são problemas de optimização, isto é, problemas em que o objectivo consiste em chegar a um programa de produção com vista à minimização e/ ou maximização de um ou mais critérios de optimização. Contudo, alguns deles são originalmente formulados numa versão mais simples de decisão, em que, numa abordagem mais simplificada, se pretende apenas obter respostas mais simples para estes problemas, do tipo fazer ou não determinados trabalhos em cada processador do sistema de produção.

Os problemas de escalonamento pertencem a uma classe muito mais ampla de problemas de optimização combinatorial que, geralmente, são difíceis de resolver, isto é, são problemas NP-difíceis ou *NP-hard*, que têm subjacente um conjunto mais ou menos vasto de possíveis combinações de soluções a explorar no espaço de soluções possíveis dos problemas (Ceponkus, 1999; Jordan, 1996; Blazewicz, 1996 e Brucker, 1995).

A título de exemplo, programar, sem restrições, n lotes em m processadores numa linha ou oficina, se fizéssemos a enumeração completa de programas, equivaleria a ter $(n!)^m$ sequências alternativas, isto é, programas para avaliar. Assim, se m=2 e n=6 o número N_S

de sequências possíveis seria $N_S = (6!)^2$, portanto $N_S = 518\ 400$ sequências e se m = 3, então $N_S = 373\ 248\ 000$ sequências!

3.6.1 Complexidade temporal

A análise da complexidade indica se é possível resolver um determinado problema de escalonamento, num intervalo de tempo superiormente limitado por um polinómio em função do comprimento de entrada do problema, isto é, dos dados do problema, ou seja, em tempo polinomial. A complexidade depende da ordem de complexidade resultante da análise do pior caso da função de complexidade e da aplicação particular em causa.

Definição 3.37 - Classe de problemas P: consiste em todos os problemas de decisão que poderão ser resolvidos através da máquina determinística de *Turing*[1], um modelo abstracto de computação, em tempo superiormente limitado por um polinómio do comprimento de entrada (Blazewicz, 1996). Sabe-se que $P \subseteq NP$.

Definição 3.38 - Classe de problemas NP: consiste em todos os problemas de decisão que possam ser resolvidos em tempo polinomial por uma máquina não determinística de *Turing*[2]. Esta Classe de problemas inclui duas subclasses genéricas de problemas que são as dos problemas do tipo *NP-hard* e do tipo *NP-complete* (Blazewicz, 1996).

Muitos dos problemas reais são problemas "difíceis", ou seja, problemas do tipo *NP-hard*. Exemplos destes problemas são a grande maioria dos problemas que ocorrem em sistemas com ambientes complexos, tais como, oficinas e sistemas flexíveis. Contudo, independentemente da natureza do sistema de produção em que o problema ocorre, é possível surgirem problemas "difíceis" de resolver na prática, mesmo em sistemas relativamente mais simples, do tipo processador único, processadores paralelos ou linhas de produção, por exemplo. A título de exemplo poderá referir-se o problema de dimensionamento de lotes ou *lot sizing* e sequenciamento de conjuntos ou problema *batch sequencing problem* (*BSP)* com tempos de preparação, que é um problema do tipo *NP-hard* (Jordan, 1996).

Esta complexidade dos problemas de escalonamento é, normalmente, agravada na presença de problemas dinâmicos (PED) que, ocorrendo em ambientes reais, possuem uma complexidade adicional face aos problemas estáticos. Em muitos casos estes problemas são difíceis de resolver, ou seja, o tempo requerido para encontrar uma solução óptima para o

[1] *Deterministic Turing Machine (DTM)*
[2] *Non-Deterministic Turing Machine (NDTM)*

problema aumenta exponencialmente com a dimensão ou dados de entrada deste (isto é, são do tipo NP-difícil ou *NP-hard*), portanto, mesmo perante situações aparentemente simples, nomeadamente em ambiente de processador único, ou linhas de produção simples.

3.6.2 Complexidade face ao tipo de ambiente de produção

Quando existem n trabalhos para serem sequenciados num sistema de processador único ou em sistemas de processadores paralelos ou numa linha pura, com uma única fila de trabalhos para ordenar, então existem n! soluções, isto é, sequências possíveis.

A complexidade do escalonamento depende de vários factores, nomeadamente, do tipo de sistema de produção em que o problema ocorre. Muito embora seja difícil ou até mesmo impossível relacionar essa complexidade em função do tipo de sistema de um modo exacto, uma forma simples e aproximada de abordar esta questão é a que se ilustra na Figura 3.18.

Da análise da Figura 3.17, pode concluir-se que o esforço requerido, a nível de escalonamento, é normalmente menor em sistemas do tipo processador único e, geralmente, "gradualmente" maior à medida que se progride de um sistema desta natureza para um sistema mais "complexo", por exemplo, um sistema flexível, nomeadamente, que produz uma grande variedade de produtos, isto admitindo que as restantes condições a influenciar o problema se mantêm para todos os casos, ou seja, que todos os problemas são semelhantes, em termos das restantes características e o único aspecto que muda é o tipo de sistema em que os trabalhos decorrem.

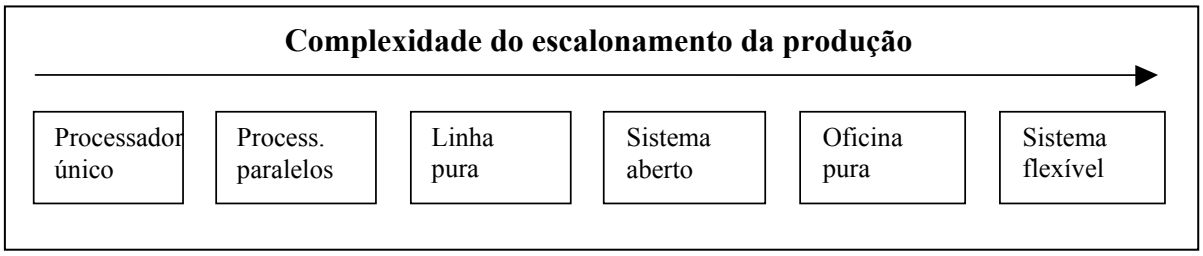

Figura 3.17 - Variação da complexidade do escalonamento com o tipo de sistema de produção.

Mesmo assim, haverá muitas situações em que a complexidade do problema não evolui exactamente da forma ilustrada, visto que cada caso é um caso e, mesmo dentro de um determinado tipo de sistema poderá verificar-se uma maior ou menor variedade de situações, ou seja, as instâncias de problemas ou antes, de sistemas, podem apresentar características mais ou menos parecidas, em termos da configuração e funcionamento do sistema de produção, ou variar consideravelmente. Sendo assim, a diferença, a nível de

complexidade do problema, poderá ser muito ténue, desaparecer completamente, ou até mesmo inverterem-se as situações de complexidade, por exemplo, um problema que ocorre numa linha pode ser menos complexo do que um outro que ocorre num sistema de processadores paralelos.

3.6.3 Complexidade face às características do problema

Os problemas de escalonamento são muito diversos e podem revelar-se de resolução muito complexa. Para dar uma ideia dessa diversidade e complexidade podem referir-se, por exemplo, os seguintes parâmetros para classificação destes problemas, de acordo com as características dos trabalhos e dos processadores e recursos auxiliares, anteriormente descritas, nomeadamente, tipo e o número de processadores que integram o sistema produtivo; quantidade de trabalhos ou operações a realizar; produção em lotes ou unitária; cumprimento de prazos de entrega; capacidade limitada dos processadores e existência de gargalos de estrangulamento no sistema; existência de armazéns locais com capacidade também limitada; possibilidade de interrupção dos trabalhos e das operações; além da existência de uma enorme variedade de critérios para avaliar o desempenho do sistema de produção; existência de restrições tecnológicas ou de precedência entre trabalhos ou operações; necessidade de preparação dos processadores, podendo esta preparação ser dependente da ordem de lançamento dos trabalhos ou operações; chegadas dinâmicas dos lotes, trabalhos ou operações ao sistema e disponibilidade limitada dos processadores.

Estes parâmetros dão origem a uma grande variedade de "categorias" ou "tipos de problemas", nomeadamente, problemas conhecidos como pertencendo às categorias de "processador único"; "processadores paralelos"; "linhas com 2 processadores"; "linhas com 3 ou mais processadores"; "oficina de n trabalhos em m processadores"; "células"; "programação de 2 trabalhos numa oficina com m processadores"; "trabalhos com restrições de precedência"; "lotes independentes"; "problema do caixeiro viajante - PCV"; "chegadas estáticas dos trabalhos ao sistema"; " processadores com disponibilidade dinâmica".

3.6.4 Complexidade face ao critério de optimização

A solução de um problema de escalonamento é determinada na base de um critério de eficiência ou de desempenho, directa ou indirectamente relacionado com variáveis tais como o prazo de fabrico, o nível de trabalhos em curso, a utilização do equipamento, a

urgência e outros. Existe uma enorme variedade de critérios para avaliar o desempenho do sistema de produção, como se constatou anteriormente, na secção 3.4.

Encontrar um bom critério de optimização, que traduza o objectivo a atingir, pode ser uma tarefa difícil no escalonamento, por várias razões. Em primeiro lugar, tais objectivos, como é o caso da satisfação do cliente com elevados níveis de qualidade no atendimento, a nível de satisfação dos seus requisitos e a prontidão, são difíceis de quantificar. Em segundo lugar, uma empresa aborda, normalmente, três tipos de objectivos que são bastante diferentes e por vezes até incompatíveis, isto é, a melhoria de um traduz-se na deterioração de outro:

- maximizar a cadência ou fluxo de produção, num determinado período de tempo;
- satisfazer os requisitos dos clientes, em termos de qualidade e cumprimento dos prazos;
- minimizar custos correntes.

3.6.5 Complexidade face à natureza das variáveis

Os problemas de escalonamento podem ser determinísticos ou estocásticos. Enquanto que num problema determinístico os valores das variáveis são conhecidos e bem definidos à partida, isto é, definidos com precisão, no caso das variáveis terem distribuições probabilísticas ou serem estocásticas o mesmo já pode não acontecer. Neste caso, as variáveis poderão assumir valores incertos, na medida em que pode não existe um único valor possível para cada variável. Assim, podem assumir-se valores de um intervalo ou conjunto de valores possíveis obtidos, por exemplo, por uma distribuição de valores probabilísticos de ocorrência.

Blazewicz (1996) afirma que a programação determinística implica, tipicamente, conhecer, à partida, os instantes de chegada e um valor do tempo de processamento de cada trabalho.

Os problemas determinísticos e probabilísticos ou estocásticos poderão ser definidos, segundo Blazewicz (1996), como se segue.

Definição 3.39 - Problema determinístico: é um problema em que as variáveis do problema são de natureza determinística, isto é, os valores de todas as variáveis do problema são conhecidos ou estimados à *priori*, nomeadamente, instantes de chegada e tempos de processamento dos trabalhos ou operações e são bem definidos e invariáveis.

Definição 3.40 - Problema estocástico: é um problema em que, ao contrário do que se passa num problema determinístico, a natureza das variáveis é estocástica ou probabilística. Assim, pelo menos uma das variáveis relevante ao resultado do escalonamento é não determinística.

3.7 Considerações finais

Na presença de problemas do tipo NP-difíceis (ou *NP-hard*), podem tentar-se relaxar algumas das restrições impostas no problema original e depois resolver o problema simplificado. A solução deste último poderá constituir uma boa aproximação para a solução do problema original.

Uma forma típica de simplificar a actividade de escalonamento consiste em simplificar o sistema de produção, à custa da "redução" ou conversão de um determinado sistema de produção num outro, para o qual a solução seja mais fácil de obter. Uma forma de conseguir esta simplificação consiste em agrupar os processadores, no intuito, por exemplo, de transformar um determinado sistema do tipo sistema de fases múltiplas em linha pura, num sistema de fase única e processador único. Neste último sistema as únicas questões que se levantam em cada instante prendem-se com a questão principal: "Qual a ordenação dos trabalhos para execução no processador?" que, de um modo mais detalhado e admitindo que não é permitida a interrupção dos trabalhos, se traduz nas seguintes questões:

- Qual o próximo trabalho a ser iniciado ou lançado em fabrico;
- Qual o instante de início do trabalho (libertação para a produção de um trabalho no processador em causa);
- Qual o instante de conclusão do trabalho (libertação do processador para o trabalho que acabou de ser processado).

Neste exemplo, como existe um único processador para realizar os trabalhos, não se coloca o problema da afectação dos trabalhos, mas apenas é equacionado o problema de sequenciamento. A calendarização é uma consequência da resolução das questões anteriormente colocadas.

Uma outra forma de simplificação poderá consistir em:

- Permitir interrupção, ainda que o problema original aborde programas sem interrupção.
- Assumir trabalhos de comprimento/ duração unitária, quando eram considerados trabalhos com durações arbitrárias no problema original.

Assim, poder-se-á optar por métodos de escalonamento menos complexos, por exemplo, no contexto de processadores paralelos, quando o objectivo consiste em minimizar o *makespan*, se for considerada a possibilidade de interrupção dos trabalhos, de modo que o processamento destes possa ser posteriormente retomado, para ser concluído no mesmo ou num outro processador, então poder-se-á usar o método de McNaughton (1959), que garante a obtenção de uma solução óptima, para um qualquer número de trabalhos, num qualquer número de processadores. O mesmo já não acontece se não for possível interromper o processamento dos trabalhos, pois o problema torna-se consideravelmente mais complexo à medida que a quantidade de trabalhos a escalonar e/ ou o número de processadores no sistema aumenta. Assumir determinados tipos de grafos de precedências, por exemplo, árvores ou caminhos, em vez de grafos arbitrários, considerados no problema original, também é uma possível forma de simplificar um problema.

Uma outra forma de tentar reduzir a complexidade do escalonamento consiste em tentar simplificar o problema em causa através da utilização de uma estratégia adequada para abordar os objectivos a atingir, em termos de especificação da forma mais adequada de definir o critério de optimização para avaliação do funcionamento do sistema de produção em causa.

A minimização do atraso máximo ou *maximum lateness* (L_{max}) é importante quando se trata de um problema relativamente simples e pode ser um ponto de partida útil para a resolução de outros problemas mais complexos, por exemplo, minimizar o tempo de percurso máximo (F_{max}). Este critério pode ser utilizado em conjugação com a consideração de datas de entrega. Minimizar o atraso máximo (L_{max}) é um critério importante quando os clientes só toleram pequenos ou ligeiros atrasos na entrega das encomendas. Este critério pode ser utilizado em simultâneo com a anterior (F_{max}) e considerado igual a zero se o valor for negativo.

Finalmente, existe um motivo ou razão técnica para usar este critério, na resolução de problemas complexos do tipo "problemas de recursos múltiplos" ou *multi-resource problem* com um objectivo do tipo "tempo máximo ou total de percurso" ou *makespan*. Se se arbitrar uma data de entrega comum para todos os trabalhos no problema, então minimizar o atraso máximo (L_{max}) será equivalente a minimizar o *makespan* (C_{max}).

Para tentar resolver eventuais situações de conflito entre critérios de optimização várias abordagens são possíveis (Ribeiro, 1999):

- resolver sub-problemas com um objectivo de cada vez;
- resolver o problema através de curvas de relacionamento entre objectivos;
- combinar vários objectivos num único objectivo mais complexo.

Sendo assim, poderá optar-se pela resolução de subproblemas, isto é, problemas com um objectivo de cada vez. O que aliás constitui uma abordagem típica à maioria dos problemas complexos reais.

Outra abordagem possível consistirá em resolver os problemas com base em curvas que relacionam diferentes objectivos, para tentar avaliar o impacto de cada objectivo nos resultados obtidos e decidir, em função da análise efectuada (Ribeiro, 1999). Ou ainda, poderão combinar-se objectivos, por exemplo, custos baixos e elevadas taxas de utilização do sistema (Ribeiro, 1999).

Uma sequência óptima de execução de lotes pode ser obtida por selecção da melhor sequência de entre todas as possíveis, obtidas por enumeração completa. Este procedimento, mesmo para problemas de dimensão reduzida pode tornar-se impraticável, tal como se pode constatar pelo exemplo anteriormente apresentado no início desta secção. Assim, torna-se muitas vezes imprescindível recorrer a outras formas, menos laboriosas que a enumeração completa, para determinar uma solução aceitável para a ordenação dos trabalhos, nomeadamente através da utilização de métodos matemáticos heurísticos ou meta-heurísticos ou através de regras de sequenciamento ou de quaisquer heurísticas simples de despacho. Estes procedimentos podem, em alguns casos, gerar soluções óptimas noutros, porém, apenas geram soluções geralmente consideradas boas ou aceitáveis. Estes e outros métodos foram anteriormente descritos no capítulo 2, na secção 2.4, e alguns métodos são ilustrados mais adiante, no capítulo 6, na secção 6.4, para a resolução de alguns problemas clássicos de escalonamento da produção.

Na literatura existem diferentes nomenclaturas para a classificação de problemas de escalonamento e no capítulo seguinte apresenta-se uma breve revisão de algumas dessas nomenclaturas principais, apresentadas por alguns autores e uma nomenclatura proposta.

4 Classificação de Problemas de Escalonamento da Produção

4.1 Introdução

Os problemas de escalonamento possuem um vasto conjunto de características, relacionadas com os trabalhos e os processadores, como se constatou pelo anteriormente exposto no capítulo 3. Estas características têm de ser definidas e especificadas para cada problema a resolver. Desta forma, é vantajoso recorrer a uma nomenclatura para a classificação dos problemas, de modo a facilitar a sua especificação e representação.

Neste capítulo apresenta-se uma revisão de nomenclaturas de representação de problemas de escalonamento e uma nomenclatura proposta, que visa uma classificação clara e abrangente de problemas de escalonamento, que permita objectivar a caracterização destes problemas com vista à identificação e selecção de métodos adequados à sua resolução.

4.2 Revisão de nomenclaturas da literatura

As nomenclaturas apresentadas a seguir são propostas por diversos autores, como Conway, French, Brucker, Blazewicz, Pinedo e Jordan.

4.2.1 Nomenclatura de Conway

Conway (1967) classifica os problemas de escalonamento com base numa nomenclatura de classificação com os seguintes parâmetros, A / B / C / D e representa-os por:

Parâmetro A

O parâmetro *A* descreve o processo de chegada dos trabalhos ao sistema.

Este parâmetro pode assumir um valor inteiro positivo ou o valor n, no caso de representar um problema em que possa existir um número arbitrário, mas finito, de trabalhos, para problemas estáticos. No caso de problemas dinâmicos, *A* identificará a distribuição dos tempos entre chegadas de cada trabalho.

Parâmetro B

Este parâmetro descreve o número de processadores no sistema e pode assumir um valor inteiro positivo, ou o valor m, para um número arbitrário de processadores.

Parâmetro C

O parâmetro *C*, que pode assumir os valores ∅, F, R e G, descreve o sistema de produção, como se segue: Sistema de processador único (e este parâmetro é omitido).

- C=∅: máquina única.
- *C* =F: linha pura.
- *C* =R: oficina geral.
- *C* =G: sistema aberto.

Parâmetro D

Este parâmetro descreve o critério de optimização através do qual os programas de produção serão avaliados. Conway ilustra critérios relacionados com a utilização dos processadores e com os trabalhos a produzir. Através de um parâmetro *D* com os valores U_{med}, N_{med}, P_{med} e C_{max} ou F_{max}, onde:

- D=U_{med}: utilização média dos processadores.
- D=N_{med}: número médio de trabalhos no sistema.
- D=P_{med}: conteúdo médio de trabalhos no sistema.
- D=C_{max}: instante máximo de conclusão dos trabalhos.
- D=F_{max}: período máximo de permanência do último trabalho a ser processado ou tempo máximo de em curso dos trabalhos em produção.

Resumo da nomenclatura A / B / C / D

Os parâmetros subjacentes a este nomenclatura de Conway e correspondentes valores que estes podem assumir estão ilustrados na Tabela 4.1.

Tabela 4.1 - Nomenclatura de Conway.

Parâmetro	Designação	Valor
A	Número de trabalhos, no caso dos problemas estáticos ou a filosofia de chegadas em problemas dinâmicos	Qualquer inteiro positivo ou n ou uma distribuição probabilística de chegadas
B	Número de processadores no sistema	Qualquer inteiro positivo ou m
C	Tipo de sistema de produção: sistema de processador único, linha pura, sistema aberto e oficina geral	∅, F, G, R
D	Critério de optimização	U_{med}, N_{med}, P_{med}, C_{max}, F_{max}

Exemplos de representação de classes de problemas:

Ex1: *n*/2/*F*/F_{max}, sequenciar um número arbitrário de trabalhos numa linha pura com dois processadores, de modo a minimizar o tempo máximo de percurso desses trabalhos ("problema de Johnson").

Ex2: *n*/*m*/*R*/F_{max}, programar *n* trabalhos, numa oficina geral de *m* processadores, de modo que o último trabalho esteja concluído o mais cedo possível.

4.2.2 Nomenclatura de French

French (1982), usa uma nomenclatura de classificação bastante similar à de Conway com um esquema de representação também baseado em quatro parâmetros de classificação: n/m/A/B.

French recorreu a esta notação simples, para representar os tipos de problemas que ocorrem em sistemas de produção.

Parâmetro n

O parâmetro n pode assumir qualquer valor inteiro positivo e o próprio valor n. Este parâmetro representa o número de trabalhos a programar e é usada a letra n para representar um intervalo não especificado ou qualquer quantidade de trabalhos.

Parâmetro m

O parâmetro m pode assumir qualquer valor inteiro positivo ou a letra m. Este parâmetro refere-se ao número de processadores no sistema de produção.

Parâmetro A

Este parâmetro descreve o sistema de produção. Quando m=1, *A* é deixado em branco.

O parâmetro *A* pode assumir valores tais como: ∅, F, P e G:

- *A* =F: linha pura, em que a ordem dos processadores é a mesma para todos os trabalhos.
- *A* =P: linha pura de permutação (*permutation flow shop*): linha em que todos os trabalhos têm o mesmo número de operações e a ordem de visita aos processadores é a mesma para todos os trabalhos, isto é, todos os trabalhos fazem a primeira operação no primeiro processador da linha, a segunda operação no segundo processador e assim sucessivamente, até à última operação de cada trabalho, no último processador da linha. Além disso, a ordem de execução dos trabalhos mantém-se em cada processador. Deste modo, um programa fica completamente especificado por uma simples permutação dos trabalhos 1, 2, …, n, dando origem à ordem pela qual estes deverão ser processados, em cada um dos processadores da linha.
- *A* =G: oficina geral.

Parâmetro B

O parâmetro *B* descreve o critério de optimização através do qual o programa é avaliado.

- *B*= F_{max}: tempo de percurso máximo.
- *B*=C_{max}: instante máximo de conclusão.
- *B*=F_{med}: tempo médio de percurso dos trabalhos.
- *B* =C_{med}: tempo médio de conclusão dos trabalhos.
- *B* =L_{med}: atraso médio dos trabalhos.
- *B*= L_{max}: atraso máximo dos trabalhos.
- *B* =T_{med}: atraso médio positivo dos trabalhos.
- *B*= N_j (N_{wmed}, N_{umed}, N_{cmed} e N_{pmed}): número de trabalhos atrasados e, respectivamente, número médio pesado de trabalhos à espera de serem processados, não acabados, concluídos e a serem processados.
- *B*=I_{med}: tempo médio de inactividade dos processadores.
- *B*=I_{max}: tempo máximo de inactividade dos processadores.

Resumo da nomenclatura n / m / A / B

Os parâmetros subjacentes a este nomenclatura de French e correspondentes valores que estes podem assumir são sumariados na Tabela 4.2.

Tabela 4.2 - Nomenclatura de French.

Parâmetro	Designação	Valor
n	Número de trabalhos	Qualquer inteiro positivo ou *n*
m	Número de processadores no sistema	Qualquer inteiro positivo ou *m*
A	Tipo de sistema de produção (Sistema de processador único, linha pura, linha pura de permutação e oficina geral)	∅, *F*, *P* e *G*
B	Critério de optimização do sistema	F_{max}, C_{max}, F_{med}, C_{med}, L_{med}, L_{max}, T_{med}, T_{max}, N_j, N_{wmed}, N_{umed}, N_{cmed}, N_{pmed}, I_{med} e I_{max}

Exemplos de representação de classes de problemas:

Ex1: n/2/F/$C_{max,}$ é um problema com n trabalhos e 2 processadores numa linha pura, onde o objectivo consiste em minimizar o tempo máximo de percurso dos trabalhos.

Ex2: *n*/*m*/*G*/F_{max}, programar *n* trabalhos, numa oficina geral de *m* processadores, de modo que o último trabalho seja concluído o mais cedo possível.

4.2.3 Nomenclatura de Brucker

Peter Brucker (1995) apresenta uma nomenclatura classificativa geral do tipo α| β| γ, cuja notação foi inicialmente introduzida por Graham et al., em 1979. Esta nomenclatura ainda pode decompor-se na seguinte nomenclatura mais detalhada: α1 α2| β1 β2 β3 β4 β5 β6| γ.

A *string* α contém características relativas ao ambiente de produção e é constituída por dois parâmetros, α1 e α2.

As características dos trabalhos e dos processadores são especificadas por um conjunto de parâmetros β, contendo no máximo 6: β1, β2, β3, β4, β5 e β6.

γ é a classe referente ao critério de optimização a considerar em cada problema.

Classe de parâmetros α

A *string* α contém características relativas ao ambiente de produção e é constituída por dois parâmetros, α1 e α2.

Valores possíveis para α1 são: ∅, P, Q, R, PMPM, QMPM, PMPT, QMPT, RMPT, G, X, O, J e F, GMPM, XMPM, OMPM, JMPM e FMPM, GMPT, XMPT, OMPT, JMPT e FMPT.

Se α1∈{∅, P, Q, R, PMPM, QMPM, PMPT, QMPT, RMPT}, onde ∅ representa o símbolo vazio (tal que: α=α2 se α1=∅), então cada trabalho J_i consiste numa única

operação O e existe um único processador ou processadores paralelos para realizar esses trabalhos, isto é, cada trabalho pode ser executado em cada um dos processadores M_1, …, M_m, no caso de existir um sistema com dois ou mais processadores paralelos:

- $\alpha 1 = \varnothing$: sistema de processador único.
- $\alpha 1$=P: sistema de processadores paralelos idênticos.
- $\alpha 1$=Q: sistema de processadores paralelos uniformes.
- $\alpha 1$=PMPM e $\alpha 1$=QMPM: sistema (de conjuntos) de processadores paralelos idênticos e uniformes (*multi-purpose identical* (*uniform*) *parallel machines*).
- $\alpha 1$=PMPT, $\alpha 1$=QMPT e $\alpha 1$=RMPT: sistema de processadores paralelos idênticos, uniformes e não relacionados de trabalhos multi-processador (*identical* (*uniform*/ *unrelated*) *parallel machines for multi-processor tasks*).

Se $\alpha 1 \in \{G, X, O, J, F\}$ estar-se-á perante um ambiente de produção clássico de multi-fase ou multi-operação, em que:

- $\alpha 1$=G: sistema geral.
- $\alpha 1$=J: oficina clássica ou pura.
- $\alpha 1$=F: linha pura.
- $\alpha 1$=O: sistema aberto.
- $\alpha 1$=X: sistema misto, neste caso, uma combinação de uma oficina pura com um sistema aberto.

Estes sistemas também podem ser generalizados de modo a incluir conjuntos de máquinas em cada estágio do sistema dando origem aos correspondentes sistemas $\alpha 1 \in \{$GMPM, XMPM, OMPM, JMPM, FMPM$\}$:

- $\alpha 1$=GMPM: sistema geral flexível.
- $\alpha 1$=JMPM: oficina clássica ou pura flexível.
- $\alpha 1$=FMPM: linha pura flexível.
- $\alpha 1$=OMPM: sistema aberto flexível.
- $\alpha 1$=XMPM: sistema misto flexível.

Os sistemas acima podem ainda ser generalizados de modo a considerar a produção de trabalhos multi-processador (*multi-processor task*, MPT) dando origem aos correspondentes sistemas $\alpha 1 \in \{$GMPT, XMPT, OMPT, JMPT, FMPT$\}$:

- $\alpha 1$=GMPT: sistema geral de trabalhos multi-processador.
- $\alpha 1$=JMPT: oficina clássica ou pura de trabalhos multi-processador.
- $\alpha 1$=FMPT: linha pura de trabalhos multi-processador.
- $\alpha 1$=OMPT: sistema aberto de trabalhos multi-processador.
- $\alpha 1$=XMPT: sistema misto de trabalhos multi-processador.

Se $\alpha 2$ é igual a um inteiro positivo 1, 2, … então representa o número efectivo de processadores.

- $\alpha 2=\varnothing$: número de processadores é variável ou arbitrário.
- $\alpha 2$=k: número fixo de processadores.

Classe de parâmetros β

As características dos trabalhos e dos processadores são especificadas por um conjunto β, contendo no máximo 6 parâmetros: β1, β2, β3, β4, β5 e β6.

β1 indica se a interrupção do processamento dos trabalhos (*preemption*) é permitida ou não.

- $\beta 1=\varnothing$: não é permitida a interrupção dos trabalhos.
- β1= pmtn: é permitida a interrupção dos trabalhos.

β2 descreve as relações de precedência entre trabalhos.

Problemas de escalonamento podem possuir restrições de precedência expressas por caminhos, árvores ou uma série de grafos direccionados paralelos. Nestes casos coloca-se β2 respectivamente igual a chains, tree e sp-graph (Brucker, 1995).

- $\beta 2=\varnothing$: não existem restrições de precedência, ou seja, β2 não figura em β.
- β2=prec: um grafo direccionado acíclico.
- β2=tree: uma árvore.
- β2=chains: um conjunto de caminhos.
- β2=sp-graph: um grafo (*series parallel- graph*).

β3 especifica as chegadas dos trabalhos (datas de lançamento) para cada trabalho.

- $\beta 3=\varnothing$ então o problema é estático e β3 não aparece em β.
- $\beta 3=r_i$ então o problema é dinâmico, sendo r_i a data de chegada do trabalho i ao sistema.

β4 especifica restrições nos tempos de processamento ou no número de operações.

- $\beta 4=\varnothing$: os tempos de processamento são arbitrários.
- $\beta 4=p_i=1$ ($p_{ij}=1$): cada trabalho (operação) tem um tempo de processamento unitário.

Ocasionalmente o campo β4 contém características adicionais com uma interpretação clara, por exemplo, $p_j \in \{1,2\}$.

β5 indica a existência de prazos de entrega para os trabalhos.

- $\beta 5=\varnothing$: não existem prazos.
- $\beta 5=d_i$: é especificada uma data d_i devida de conclusão de cada trabalho J_i.

Em algumas aplicações de escalonamento, os conjuntos de trabalhos terão de ser agrupados em lotes e esta característica é representada pelo parâmetro β6.

- $\beta 6=\varnothing$: não existe produção em lotes e este parâmetro será omitido na estrutura classificativa do problema.
- β6=batch: um problema em que a estrutura de produtos são lotes.

Classe do parâmetro γ

γ é o parâmetro da classe referente ao critério de optimização a considerar em cada problema.

Este autor categoriza quatro tipos de objectivos: $\gamma = \max G_i$, $\max w_i G_i$, $\sum G_i$ e $\sum w_i G_i$.

Um dos objectivos que este autor considera mais importantes, além de C_{max}, é $L_{max}:=\max(i=1 \text{ a } n)\ L_i$ (atraso máximo do trabalho i).

Uma lista mais alargada de critérios possíveis inclui: $\gamma \in \{C_{max}, \sum C_i, \sum w_i C_i, \max G_i, \max w_i G_i, \sum G_i, \sum w_i G_i, L_{max}, \sum T_i, \sum w_i T_i, \sum U_i, \sum w_i U_i, \sum D_i, \sum w_i D_i, \sum S_i, \sum w_i S_i, \sum E_i, \sum w_i E_i\}$.

Resumo da nomenclatura α | β | γ

As classes, com os respectivos parâmetros subjacentes a este nomenclatura de Brucker e correspondentes valores que estes podem assumir são sumariados na Tabela 4.3.

Tabela 4.3 - Nomenclatura de Brucker.

Classe	Parâmetro	Designação	Valor
α	$\alpha 1$	Tipo de sistema de produção: sistemas uni-operação: sistema de processador único, sistemas de processadores paralelos (idênticos, uniformes e não relacionados), sistema flexível de processadores paralelos (idênticos ou uniformes). Sistemas multi-operação: sistema geral, sistema misto, sistema aberto, oficina clássica ou pura e linha pura e correspondentes sistemas flexíveis (MPM) e sistema de trabalhos multi-processador (MPT)	∅, *P*, *Q*, *R*, *PMPM*, *QMPM*, *PMPT*, *QMPT*, *RMPT*, *G*, *X*, *O*, *J* e *F*, *GMPM*, *XMPM*, *OMPM*, *JMPM* e *FMPM*,*GMPT*, *XMPT*, *OMPT*, *JMPT* e *FMPT*
	$\alpha 2$	Número de processadores no sistema	∅, um valor inteiro positivo, *k*
β	$\beta 1$	Interrupção	∅, pmtn
	$\beta 2$	Restrição de precedência	prec, chain, intree, outtree, sp-graph
	$\beta 3$	Filosofia de chegadas	∅, r_i
	$\beta 4$	Restrições nos tempos de processamento	$p_i=1$, $p_{ij}=1$, $p_i \in \{1,2\}$
	$\beta 5$	Prazos	∅, d_i
	$\beta 6$	Lotes	∅, batch
γ	γ	Critério de optimização do sistema	C_{max}, $\sum C_i$, $\sum w_i C_i$, $\max G_i$, $\max w_i G_i$, $\sum G_i$, $\sum w G_i$, L_{max}, $\sum T_i$, $\sum w_i T_i$, $\sum U_i$, $\sum w_i U_i$, $\sum D_i$, $\sum w_i D_i$, $\sum S_i$, $\sum w_i S_i$, $\sum E_i$, $\sum w_i E_i$

Exemplos de representação de classes de problemas:

Ex1: $P|prec; p_i=1|C_{max,}$ é um problema estático de escalonamento de trabalhos, com uma única operação, com tempos de processamento unitários e restrições gerais de precedência entre trabalhos, num sistema de m processadores paralelos idênticos, em que se pretende que o tempo de conclusão máximo desses trabalhos seja minimizado.

Ex2: $|batch|\sum w_i C_{i,}$ é o problema de escalonamento em que se considera um conjunto de trabalhos agrupados em lotes, a programar num sistema com um único processador, de modo a minimizar o tempo de percurso pesado destes trabalhos no sistema.

4.2.4 Nomenclatura de Blazewicz

Blazewicz propõe uma nomenclatura classificativa semelhante à de Brucker, com 11 parâmetros. Este autor ainda expande esta nomenclatura para incluir outros parâmetros importantes no processo de classificação de problemas em que os trabalhos a escalonar são tarefas em computadores. Estes parâmetros adicionais não serão aqui apresentados, uma

vez que não se inserem no âmbito deste trabalho, que visa apenas o estudo de problemas de produção industrial.

O nomenclatura de classificação de Blazewicz (1996) também é uma nomenclatura que integra três classes fundamentais de parâmetros, $\alpha \mid \beta \mid \gamma$ que, na forma mais detalhada, pode ser representada por: $\alpha 1\ \alpha 2 \mid \beta 1\ \beta 2\ \beta 3\ \beta 4\ \beta 5\ \beta 6\ \beta 7\ \beta 8 \mid \gamma$.

A primeira classe, α, descreve o ambiente de produção e integra os parâmetros $\alpha 1$ e $\alpha 2$.

O parâmetro $\alpha 1$ caracteriza o tipo de processadores usados e o parâmetro $\alpha 2 \in \{\varnothing, k\}$ representa o número de processadores no sistema. A segunda classe, β, descreve as características dos trabalhos, das operações e dos processadores, de $\beta 1$ a $\beta 8$. A terceira classe integra um parâmetro γ, que expressa um critério de optimização.

Classe de parâmetros α

O parâmetro relativo ao tipo de sistema de produção é $\alpha 1 \in \{\varnothing, P, Q, R, O, F, J\}$ em que:

- $\alpha 1 = \varnothing$: sistema de processador único (nota: nesta notação $\varnothing$ representa um símbolo vazio que é omitido nos problemas deste tipo).
- $\alpha 1 = P$: sistema de processadores paralelos idênticos.
- $\alpha 1 = Q$: sistema de processadores paralelos uniformes.
- $\alpha 1 = R$: sistema de processadores paralelos não relacionados.
- $\alpha 1 = O$: sistema aberto.
- $\alpha 1 = F$: linha pura.
- $\alpha 1 = J$: oficina pura.

Parâmetro $\alpha 2 \in \{\varnothing, k\}$ representa o número de processadores no sistema:

- $\alpha 2 = \varnothing$: assume-se que o número de processadores é variável.
- $\alpha 2 = k$: assume-se que o número de processadores é fixo e igual a k (k é um valor inteiro positivo).

Classe de parâmetros β

A segunda classe que descreve as características dos trabalhos, das operações e dos processadores e é expressa por $\beta = \beta 1, \beta 2, \beta 3, \beta 4, \beta 5, \beta 6, \beta 7$ e $\beta 8$.

O parâmetro $\beta 1 \in \{\varnothing, pmtn\}$ indica a possibilidade de interrupção dos trabalhos:

- $\beta 1=\varnothing$: não é permitida interrupção.
- $\beta 1=pmtn$: é permitida a interrupção dos trabalhos.

O parâmetro $\beta 2 \in \{\varnothing, auxk\}$ caracteriza os recursos auxiliares.

- $\beta 2=\varnothing$: não existem recursos auxiliares.
- $\beta 2=res$: existem recursos auxiliares.

O parâmetro $\beta 3 \in \{\varnothing, prec, uan, tree, chains\}$ exprime as restrições de precedência, tais como:

- $\beta 3=\varnothing$: trabalhos independentes.
- $\beta 3=prec$: existem restrições de precedência.
- $\beta 3=uan$: redes "uni-conectadas" de actividades.
- $\beta 3=tree$: restrições de precedência em forma de árvore.
- $\beta 3=chains$: restrições de precedência na forma de um conjunto de caminhos.

O parâmetro $\beta 4 \in \{\varnothing, r_j\}$ descreve os instantes de chegada dos trabalhos ao sistema:

- $\beta 4=\varnothing$: todos os instantes de chegada são nulos.
- $\beta 4=r_j$: os instantes de chegada diferem de trabalho para trabalho.

O parâmetro $\beta 5 \in \{\varnothing, p_j=p\}$ descreve os tempos de processamento dos trabalhos:

- $\beta 5=\varnothing$: todos os trabalhos têm tempos de processamento arbitrários.
- $\beta 5=p_j$; $p_j=p$: todos os trabalhos têm tempos de processamento igual a p unidades.
- $\beta 5=p_j$; $p_{inf} \leq p_j \leq p_{sup}$: todos os trabalhos têm tempos de processamento tal que nenhum p_j é menor do que p_{inf} ou superior a p_{sup}.

O parâmetro $\beta 6 \in \{\varnothing, d_j\}$ descreve os prazos de conclusão dos trabalhos:

- $\beta 6=O$: não se assumem prazos no sistema (contudo os prazos podem ser definidos se for usado um critério de optimização envolvendo prazos, para avaliar os programas.
- $\beta 6=d_j$: são impostos prazos para a realização de um conjunto de trabalhos.

O parâmetro $\beta 7 \in \{\varnothing, n_j \leq k\}$ descreve o número máximo de operações que constituem um trabalho j.

No caso de se tratar de uma oficina pura:

- $\beta 7=\varnothing$: o número de operações é arbitrário.

- $\beta 7=(n_j \leq k)$: o número de operações, para cada trabalho não é superior a k.

O parâmetro $\beta 8 \in \{\varnothing$, no-wait$\}$ descreve uma propriedade do tipo "não-espera":

- $\beta 8=\varnothing$: é assumida a existência de armazéns locais de capacidade ilimitada.
- $\beta 8=$no-wait: os armazéns locais entre os processadores têm capacidade nula e um trabalho após ter terminado o seu processamento num processador deve iniciar, imediatamente, o seu processamento no processador seguinte.

Classe γ

A terceira classe γ representa um critério de optimização e é ilustrado como se segue, em que, por exemplo $\sum C_j$, é equivalente a considerar :

$\gamma \in \{C_{max}, \sum C_j, \sum w_j C_j, L_{max}, \sum N_{Tj}, \sum w_j N_{Tj}, \sum T_j, \sum w_j T_j, \sum E_j, \sum w_j E_j, -\}$

onde, $\sum C_j$ é equivalente a C_{med}, $\sum w_j C_j$ é equivalente a C_{wmed}, $\sum T_j$ é equivalente a T_{med}, $\sum w_j T_j$ é equivalente a T_{wmed}, $\sum E_j$ é equivalente a E_{med}, $\sum w_j E_j$ é equivalente a E_{wmed}, $\sum N_{Tj}$ é equivalente a N_{Tmed}, e $\sum w_j N_{Tj}$ é equivalente a N_{Twmed} e "-" significa testar a viabilidade do programa de modo a satisfazer os prazos ou datas de entrega impostas para os trabalhos.

Resumo da nomenclatura α | β | γ

As classes, com os respectivos parâmetros subjacentes a este nomenclatura de Blazewicz e correspondentes valores que estes podem assumir são sumariados na Tabela 4.4.

Tabela 4.4 - Nomenclatura de Blazewicz.

Classe	Parâmetro	Designação	Valor
α	$\alpha 1$	Tipo de sistema de produção: sistema de processador único, sistemas de processadores paralelos (idênticos, uniformes e não relacionados), linha pura, oficina pura e sistema aberto	∅, *P*, *Q*, *R*, F, J, *O*
	$\alpha 2$	Número de processadores no sistema	∅, *k*
β	$\beta 1$	Interrupção	∅, pmtn
	$\beta 2$	Recursos adicionais	∅, auxk
	$\beta 3$	Restrições de precedência	∅, prec, uan, tree, chains
	$\beta 4$	Filosofia de chegadas	∅, r_j
	$\beta 5$	Restrições nos tempos de processamento	∅, p_j, [p_{inf}, p_{sup}]
	$\beta 6$	Prazos	∅, d_j
	$\beta 7$	Número máximo de operações de um trabalho, no caso de oficinas	∅, [0, *k*]
	$\beta 8$	Armazéns locais	∅, no-wait (buffer)
γ	γ	Critério de optimização do sistema	C_{max}, $\sum C_j$, $\sum w_j C_j$, L_{max}, $\sum N_{Tj}$, $\sum w_j N_{Tj}$, $\sum T_j$, $\sum w_j T_j$, $\sum E_j$, $\sum w_j E_j$, "-"

Exemplos de representação de classes de problemas:

Ex1: $P||C_{max,}$ lê-se como se segue: programação de trabalhos sem interrupção e independentes, com tempos de processamento arbitrários, chegando ao sistema no instante zero, a serem executados num sistema de processadores paralelos idênticos, no intuito de minimizar o tempo máximo de conclusão dos trabalhos.

Ex2: $O3|pmtn, r_j|\sum C_j$, traduz-se por: um programa com interrupção de trabalhos, num sistema aberto com 3 processadores, onde os trabalhos chegam em diferentes instantes de tempo e o objectivo consiste em minimizar o tempo médio de percurso por trabalho.

4.2.5 Nomenclatura de Pinedo

Pinedo (2002) também apresenta uma nomenclatura para classificação de problemas de escalonamento da produção de três classes, na forma $\alpha \mid \beta \mid \gamma$, tal como Brucker e Blazewicz. Esta nomenclatura integra os seguintes parâmetros fundamentais: $\alpha| \beta1\ \beta2\ \beta3\ \beta4\ \beta5\ \beta6\ \beta7\ \beta8\ \beta9\ \beta10| \gamma$. Este autor inclui parâmetros adicionais, comparativamente às nomenclaturas anteriores, nomeadamente parâmetros relacionados com a disponibilidade e elegibilidade dos processadores, e relacionados com determinados fenómenos associados aos sistemas de produção, tais como, bloqueamentos em linhas e possibilidade de

recirculação ou permissão de reentrada de trabalhos num determinado processador de uma oficina.

A primeira classe descreve, também, o ambiente de produção e integra apenas um parâmetro α, que expressa simultaneamente o tipo de sistema e a quantidade de processadores.

A segunda classe, β, descreve detalhes relativos a características e restrições de processamento, podendo, à semelhança das nomenclaturas anteriores, conter uma única ou múltiplas entradas.

Finalmente, a terceira classe, γ, exprime um critério de optimização.

Classe de parâmetros α

A primeira classe integra então um parâmetro, α, para descrever o ambiente de produção e este autor ilustra este parâmetro com: $\alpha \in \{1, Pm, Qm, Rm, Fm, FFs, Om, Jm\}$, que caracteriza, simultaneamente, o tipo de sistema de produção e quantidade de processadores que integra:

- α=1: sistema de processador único.
- α=Pm: sistema de processadores paralelos idênticos.
- α=Qm: sistema de processadores paralelos uniformes.
- α=Rm: sistema de processadores paralelos não relacionados.
- α=Fm: linha pura.
- α=FFs: linha flexível (linha composta por processadores paralelos nos estágios).
- α=Om: sistema aberto.
- α=Jm: oficina pura.

Classe de parâmetros β

A segunda classe descreve, portanto, características e restrições de processamento, podendo incluir uma ou mais das seguintes entradas β=β1, β2, β3, β4, β5, β6, β7, β8, β9, β10.

O parâmetro $\beta 1 \in \{\varnothing, r_j\}$ descreve as chegadas dos trabalhos ao sistema:

- β1=$\varnothing$: todos os trabalhos estão simultaneamente disponíveis à partida para serem processados, i.e., todos os instantes de chegada são nulos.
- β1=r_j: os instantes de chegada diferem de trabalho para trabalho.

O parâmetro β2 expressa tempos de preparação de processadores dependentes ou não da ordem de processamento dos trabalhos, $\beta 2 \in \{\varnothing, sjk\}$:

- β2=∅: não é considerado setup ou este é incluído nos tempos de processamento dos trabalhos.
- β2=sjk: é considerado tempo de setup entre os trabalhos j e k.
- β2=sijk: é considerado setup entre os trabalhos j e k e este é dependente do processador i.

O parâmetro $\beta 3 \in \{\varnothing, pmtn\}$ indica a possibilidade de interrupção de trabalhos:

- β3=∅: não é permitido interromper trabalhos.
- β3=prmp: são permitidas interrupções nos trabalhos.

O parâmetro β4 representa a existência de restrições de precedência entre trabalhos.

$\beta 4 \in \{\varnothing, prec, chains, tree: intree, outtree\}$:

- β4=O: trabalhos independentes.
- β4=chains.
- β4=tree (intree): cada trabalho possui no máximo um sucessor.
- β4=tree (outtree): cada trabalho possui no máximo um predecessor.

O parâmetro $\beta 5 \in \{\varnothing, brkdwn\}$ indica existência de quebras na disponibilidade dos processadores, por exemplo expressas por uma qualquer função m(t):

- β5=∅: os processadores estão sempre disponíveis para realizar os trabalhos.
- β5= brkdwn: os processadores podem ter quebras de disponibilidade.

O parâmetro $\beta 6 \in \{\varnothing, elejk\}$ reflecte a existência de restrições na elegibilidade dos processadores:

- β6=∅: processadores sempre elegíveis.
- β6= elejk: possibilidade de processadores com elegibilidade restrita.

O parâmetro β7 permite identificar tipos específicos de linhas, que são as linhas de permutação (permutation flow shop). $\beta 7 \in \{\varnothing, pmtn\}$:

- β7=∅: linhas puras.
- β7=linhas de permutação.

O parâmetro β8 representa bloqueamento em linhas. Trata-se de um fenómeno de bloqueamento que pode ocorrer em linhas, quando existem buffers limitados entre processadores sucessivos da linha (*blocking*). $\beta 8 \in \{\varnothing, \text{block}\}$:

- β8=∅: linhas sem bloqueamento.
- β8= block: linhas com bloqueamento (*blocking*).

O parâmetro β9 representa a propriedade do tipo "não-espera" (no-wait), entre dois processadores sucessivos de uma linha, que uma vez activa não permite a espera de trabalhos entre processadores das linhas. $\beta 9 \in \{\varnothing, \text{nwt}\}$:

- β9=∅: são permitidas esperas entre processadores sucessivos de linhas, dada a existência de buffers intermédios.
- β9=nwt: não são permitidas esperas de trabalhos entre processadores sucessivos de linhas. Os armazéns locais ou buffers entre os processadores têm capacidade nula e um trabalho após ter terminado o seu processamento num processador deverá iniciar, imediatamente, o seu processamento no processador seguinte, podendo não iniciar, no caso de esta não estar disponível.

O parâmetro $\beta 10 \in \{\varnothing, \text{recrc}\}$ indica a possibilidade de recirculação ou reentrada de trabalhos nos processadores e é típico em oficinas:

- β10=∅: não são permitidas reentradas de trabalhos nos processadores (portanto, oficina pura).
- β10= recrc: são permitidas reentradas de trabalhos nos processadores (portanto, oficina geral).

Este autor contempla ainda a possibilidade de serem incluídos os parâmetros pj=p e dj=d na nomenclatura, para indicar tempos de processamento e datas de entrega constantes para os trabalhos, respectivamente.

Classe γ

A terceira classe γ representa um critério de optimização, que pode ser:

$\gamma \in \{C_{max}, L_{max}, T_{max}, U_{max}, \sum w_j C_j, \sum w_j(1-e^{-rC_j}), \sum w_j T_j, \sum w_j U_j$, outro critério combinado com datas de entrega$\}$.

Pinedo dá especial relevância aos critérios expressos em função dos instantes de conclusão dos trabalhos, que podem também ser expressos em função de datas de entrega.

Resumo da nomenclatura α| β| γ

As classes, com os respectivos parâmetros subjacentes a este nomenclatura de Pinedo e correspondentes valores que estes podem assumir são sumariados na Tabela 4.5.

Tabela 4.5 - Nomenclatura de Pinedo.

Classe	Parâmetro	Designação	Valor
α	α	Tipo de sistema de produção e quantidade de processadores: sistema de processador único, sistema processadores paralelos (idênticos, uniformes e não relacionados), linha pura, oficina pura, sistema aberto e linha flexível)	1, *Pm*, *Qm*, *Rm*, *Fm*, *Jm*, *Om, FFs*
β	$\beta 1$	Chegadas dos trabalhos	∅, r_j
	$\beta 2$	Tempos de preparação (*setup*)	∅, *sjk, sijk*
	$\beta 3$	Interrupção dos trabalhos	∅, *prmp*
	$\beta 4$	Restrições de precedência	∅, prec: chains, intree, outtree
	$\beta 5$	Quebras de disponibilidade nos processadores	∅, *brkdwn (ex: m(t))*
	$\beta 6$	Restrições na elegibilidade dos processadores	∅, $eleg_k$
	$\beta 7$	Linhas de permutação	∅, prmn
	$\beta 8$	Bloqueamento de linhas	∅, block
	$\beta 9$	Ausência de esperas entre processadores sucessivos (no-wait)	∅, nwt
	$\beta 10$	Recirculação em processadores em oficinas	∅, recrc
γ	γ	Critério de optimização do sistema	C_{max}, L_{max}, T_{max}, U_{max}, $\sum w_j C_j$, $\sum w_j(1-e^{-rC_j})$, $\sum w_j T_j$, $\sum w_j U_j$, ou combinações destas com datas de entrega

Exemplos de representação de classes de problemas:

Ex1: $F||C_{max,}$ lê-se como se segue: programação de trabalhos sem interrupção e independentes, com tempos de processamento arbitrários, chegando ao sistema no instante zero e executadas em linhas, no intuito de minimizar o tempo total de produção.

Ex2: $1|r_j;$ $prmp|L_{max,}$ é o problema dinâmico de encontrar um programa que permita interrupção, num processador, para um conjunto de trabalhos com determinados tempos de chegada (instantes de lançamento) rj≠0, de tal modo que o atraso máximo seja minimizado.

4.2.6 Nomenclatura de Jordan

Jordan (1996) utiliza uma nomenclatura para classificação de problemas de escalonamento expandidos, que inclui questões de loteamento (*batching*). Trata-se também de uma nomenclatura de três classes, na forma α|β|γ. Esta nomenclatura inclui os seguintes parâmetros principais: α| β1 β2 β3| γ. A primeira classe α, tal como no nomenclatura

anterior de Pinedo, inclui apenas o próprio parâmetro α, que representa as características do ambiente de produção. A classe β é restrita, contendo apenas três e eventualmente quatro parâmetros, associados aos trabalhos. Esta classe inclui parâmetros relacionados com o processamento de famílias de produtos em lotes, considerações não tidas em conta nas outras nomenclaturas. Por último, inclui a habitual classe γ, para expressar o critério de optimização.

Classe de parâmetros α

A classe α descreve então as características do ambiente de produção, podendo assumir os seguintes parâmetros: $\alpha \in \{1, F, ML1, ML, P\}$:

- α=1: sistema de processador único.
- α=F: linha flexível, com processadores paralelos idênticos (*flow-shop with identical machines*).
- α=ML1: sistema de processador único multi-função, para trabalhos complexos (produtos de multi-estrutura) (*multi-level*, *single-machine*).
- α=ML: sistema de processadores paralelos não relacionados (multi-processadores) para trabalhos complexos (produtos de multi-estrutura), com atribuição fixa de famílias a processadores (*multi-level*, *multi-machine*, *fixed family-machine assignment*).
- α=P: sistema de processadores paralelos idênticos, para trabalhos simples, de nível ou operação única (sistema uni-operação) (*identical parallel machines, single-level*).

Classe de parâmetros β

A segunda classe descreve as características dos trabalhos, tal que, β=(fam, β1, β2, β3), podendo ainda ser especificado um quarto parâmetro p(i,j)=1, para expressar que os tempos de processamento de todos os trabalhos são unitários.

O parâmetro $\beta 1 \in \{$ia-pb, ia-npb, ba, *$\}$ tipo de loteamento dos trabalhos:

- β1=ia-pb: disponibilidade dos trabalhos (lotes) com interrupção.
- β1= ia-npb: disponibilidade dos trabalhos (lotes) sem interrupção.
- β1= ba: disponibilidade de trabalhos (lotes).
- β=*: qualquer dos casos anteriores.

O parâmetro $\beta 2 \in \{$stg,i, sti, *$\}$ preparação dos processadores:

- β2= stg,i: tempos de preparação dependentes da sequência de processamento dos trabalhos.
- β2= sti: tempos de preparação independentes da sequência de processamento dos trabalhos.
- β2= *: qualquer dos casos anteriores.

O parâmetro β3 $\in$ {d(i,j), r(i,j)D, *} filosofia de chegadas e prazos de conclusão dos trabalhos:

- β3= d(i,j): cada trabalho possui um prazo (data) de conclusão devido rígido (*deadline*).
- β3= r(i,j)D: cada trabalho possui um instante de chegada e um prazo de conclusão rígido (deadline) comum é atribuído a todos os trabalhos.
- β3=*: qualquer dos casos anteriores.

Classe γ

A terceira classe γ expressa, novamente, um critério de optimização a considerar como objectivo do escalonamento e os casos apresentados por este autor são: $\gamma \in \{\sum w_{(i,j)}C_{(i,j)}, \sum sc_{g,i}, \sum sc_i, *, \}$ em que, $\sum w_{(i,j)}C_{(i,j)}$ representa a minimização dos adiantos (*earliness*) pesados, $\sum sc_{g,i}$ e $\sum sc_i$ expressa a minimização de custos de preparação, * indica a minimização da soma de adiantos e custos de preparação e " " (caso de entrada vazia ou omitida) significa que se procura encontrar um programa possível que, em geral, não é um problema trivial.

Resumo da nomenclatura α | β | γ

As classes, com os respectivos parâmetros subjacentes a esta nomenclatura de Jordan e correspondentes valores que estes podem assumir são sumariados na Tabela 4.6.

Tabela 4.6 - Nomenclatura de Jordan.

Classe	Parâmetro	Designação	Valor
α	α	Tipo de sistema de produção, com características dos trabalhos e dos processadores: sistema de processador único, linha flexível com processadores paralelos idênticos, sistema de processador único multi-função, para trabalhos complexos, sistema de processadores paralelos não relacionados para trabalhos complexos com atribuição fixa de famílias a processadores, processadores paralelos idênticos para trabalhos simples ou de nível único	1, F, ML1, ML, P
β	$\beta 1$	Tipo de loteamento dos trabalhos	ia-pb, ia-npb, ba, *
	$\beta 2$	Preparação dos processadores	stg,i, sti, *
	$\beta 3$	Filosofia de chegadas e prazos dos trabalhos	d(i,j), r(i,j)D, *
	$\beta 4$	Tempos de processamento	p(i,j)=1
γ	γ	Critério de optimização do sistema	$\sum w_{(i,j)}C_{(i,j)}$, $\sum sc_{g,i}$, $\sum sc_i$, *, “ “

Exemplos de representação de classes de problemas:

Ex1: $P||C_{max,}$ lê-se como se segue: programação de trabalhos sem interrupção e independentes, com tempos de processamento arbitrários, chegando ao sistema no instante zero, e são executadas em processadores paralelos idênticos, no intuito de minimizar o tempo total de produção.

Ex2: $1|p_{ij}=1|C_{max,}$ é o problema estático de minimizar o tempo máximo de conclusão num processador com tempos de processamento unitários.

4.2.7 Outras classificações

Os autores Baker (1974) , Vollmann e outros (1997) e Morton e Pentico (1993), entre muitos outros, recorrem à linguagem natural para classificar os diferentes tipos de problemas de escalonamento. Este tipo de classificação é o processo mais intuitivo e subjectivo de classificação, que contém grandes desvantagens, principalmente quando o objectivo da classificação consiste na obtenção de uma classificação e identificação objectiva, clara e estruturada, que permita um tratamento computacional do problema. Deste modo, será vantajoso recorrer a uma representação simbólica dos parâmetros de classificação quantitativos e qualitativos, que seja mais clara, precisa, sucinta e mais fácil de integrar numa metodologia para associação fácil de métodos a problemas, para a sua resolução, em diversos tipos de cenários industriais.

4.3 Comparação de nomenclaturas

A fim de permitir uma análise comparativa mais fácil e objectiva das várias nomenclaturas anteriormente descritas na secção 4.2, apresentam-se, a seguir, tabelas com os resumos das principais características consideradas para a classificação dos problemas de escalonamento, divididas nos dois grupos principais de características, relativas ao ambiente de produção e aos trabalhos e aos processadores e recursos auxiliares de produção.

4.3.1 Características do ambiente de produção

Na Tabela 4.7 apresenta-se um resumo das principais características relativas ao ambiente de produção, para as várias nomenclaturas anteriormente consideradas.

Tabela 4.7 - Características relativas ao ambiente de produção (parâmetros α).

Características Ambiente Prod.	Conway	French	Brucker	Blazewicz	Pinedo	Jordan	Morton	Baker	Vollmann
f1	X	X	X	X	X	X	X	X	X
f1/PI	X	X	X[*1]	X	X	X	X	X	X
f1/PU	X	X	X[*1]	X	X		X		
f1/PN	X	X	X	X	X		X		
GF/fm/F			X						
G/fm/F	X	X	X	X	X	X			X
fm/FP		X			X		X	X	X
GF/fm/O							X		X
G/fm/O					X				X
fm/OP									
G/fm							X		
GM/fm/J									
M/fm/J									X
M/fm/JP									X
GF/fm/J		X			X				
G/fm/J	X		X	X	X		X	X	X
GF/fm/JP			X						
G/fm/O	X		X	X	X		X		
GFM			X						X
Número Processadores	X	X	X	X	X		X	X	X

Da análise dessa tabela pode confirmar-se a existência de nomenclaturas com semelhanças significativas, como se viu na secção anterior, nomeadamente as nomenclaturas de French e Conway. A nomenclatura de Baker contém, basicamente, o mesmo tipo de informação,

[*1] Inclui o caso de processadores flexíveis. Do inglês, *Multi-Purpose Machines (MPM).*

contudo trata-se de uma nomenclatura descrita em linguagem natural, como se referiu anteriormente.

Além disso, as nomenclaturas de Brucker e Blazewicz também possuem grandes semelhanças.

No que se refere às nomenclaturas de Morton e de Vollmann, trata-se de nomenclaturas que recorrem à linguagem natural e ambos autores consideram uma variedade relativamente grande de tipos de sistemas de produção, além de Brucker e de Blazewicz.

4.3.2 Características dos trabalhos, dos processadores e dos recursos

A Tabela 4.8 permite uma análise semelhante à anterior, no que se refere às características dos trabalhos e dos processadores e recursos auxiliares de produção.

Tabela 4.8 - Características relativas aos trabalhos, processadores e recursos (β).

Condições no problema	Conway	French	Brucker	Blazewicz	Modelo Pinedo	Jordan	Morton	Baker	Vollmann
Número de Trabalhos	X	X		X			X		
Lotes			X			X	X		
Famílias						X			
Filosofia de chegadas	X		X	X	X	X	X	X	X
Interrupção			X	X	X	X	X		
Pesos									
Restrições de Precedência			X	X	X		X	X	
Condições nos tempos proc.			X	X	X	X	X		
Prazos			X	X	X			X	
Operações multi-processador							X		
Processadores críticos							X	X	X
Processadores agregados									
Disponibilidade dos process.					X		X	X	X
Armazéns locais				X	X		X	X	X
Preparação processadores					X	X	X	X	
Esperas nos processadores							X		
Recursos auxiliares				X					
Processadores multi-Item									
Processadores elegíveis					X				

Da análise da Tabela 4.8 pode concluir-se que os autores que consideram uma maior quantidade de parâmetros deste tipo, na classificação dos problemas são, Morton, Baker, Brucker e Blazewicz, muito embora a natureza da informação seja, por vezes, diferente.

De um modo geral, poderá dizer-se que, mais uma vez, as nomenclaturas de Brucker e Blazewicz, cobrem um conjunto similar de parâmetros e o mesmo se verifica em relação a Morton e a Baker, que apesar de recorrerem a uma classificação em linguagem natural, cobrem uma quantidade relativamente vasta de parâmetros.

Em contrapartida, as nomenclaturas de French e de Conway são bastante pobres no que diz respeito a estas características.

4.3.3 Resumo da comparação de nomenclaturas

Continuando a análise comparativa, anteriormente efectuada e, em resumo, poderá dizer-se que, embora os autores, em conjunto, acabem por conseguir focar, directa ou indirectamente, implícita ou explicitamente, numa ou noutra classe de problemas, quase todos os parâmetros anteriormente referidos e tabelados, relativamente às principais características dos trabalhos, dos processadores e dos recursos auxiliares de produção, como se verifica, através das análises anteriormente efectuadas, o que se constata é que, os autores que utilizam uma classificação descritiva, através do recurso à linguagem natural, conseguem, regra geral, focar um ou outro parâmetro fulcral para a caracterização dos problemas, relativamente a alguns dos aspectos aqui considerados importantes, embora isso nem sempre fique muito claro, porque estes não conseguem sintetizar e concentrar todas essas características num único título ou frase, necessitando por vezes de vários parágrafos! Deste modo, esses títulos que identificam os problemas ficam apenas com parte da informação, que será aquela considerada mais relevante, num determinado contexto realçado. Mas quando o objectivo é utilizar uma nomenclatura de classificação para apoiar a selecção de métodos é necessário e conveniente recorrer a uma classificação mais abrangente, além de estruturada, sintética e objectiva, de modo a permitir equacionar grande parte dos problemas que se podem identificar na realidade industrial ou, pelo menos, aqueles para os quais métodos têm sido propostos para os resolver.

Na Tabela 4.9 apresentam-se os resultados de uma avaliação e comparação mais "qualitativa" das abordagens dos diferentes autores. Esta avaliação pretende realçar, em termos gerais, as semelhanças e diferenças entre as nomenclaturas apresentadas.

Tabela 4.9 - Caracterização qualitativa dos nomenclaturas.

	Clareza e objectividade	Abrangência	Detalhe	Nomenclatura	Facilidade de integração e de utilização
Morton	E	B	B	X	F
Conway	B	R	R	B	B
Baker	E	R	R	X	F
Vollmann	E	F	F	X	F
Blazewicz	B	B	B	B	E
Brucker	B	B	B	B	E
French	B	F	F	B	F
Pinedo	B	B	B	B	E
Jordan	B	F	F	B	E

Legenda: E → Excelente B → Bom R → Razoável F → Fraco X → Não se aplica.

Dado que as nomenclaturas existentes, anteriormente descritas, não permitem cobrir um conjunto vasto de tipos de problemas de produção que se podem identificar na prática industrial houve necessidade de propor uma nomenclatura mais abrangente, que permita cobrir as principais características consideradas nas nomenclaturas apresentadas e mais algumas, nomeadamente, no que se refere a tipos de ambientes de produção, através da proposta de uma estrutura classificativa alargada. Relativamente às características dos trabalhos, processadores e recursos auxiliares existem também alguns parâmetros que em geral não são considerados pelos diversos autores, tais como, a prioridade ou pesos dos trabalhos, a existência de recursos agregados e processadores multi-item, tal como se pode observar na anterior Tabela 4.8.

Sendo assim, houve necessidade de expandir o universo dos ambientes de produção, de modo a permitir tratar problemas de escalonamento que ocorrem numa gama alargada de sistemas de produção, considerando estas características adicionais, que podem ocorrer nos diferentes sectores industriais e em serviços.

Neste trabalho propõe-se, assim, uma nomenclatura que tem em vista possibilitar uma classificação bastante exaustiva, capaz de se adequar à maioria dos problemas encontrados na teoria e prática de escalonamento.

4.4 Nomenclatura proposta

Dada a existência de uma grande variedade de problemas de escalonamento existe a necessidade de recorrer a uma nomenclatura clara e o mais possível completa de classificação de problemas que possa servir de base à identificação não ambígua e bastante

exaustiva destes. Teve-se também em conta o objectivo de utilizar uma tal nomenclatura para obtenção de conhecimento útil para a resolução destes problemas.

Houve, portanto, necessidade de especificar uma nova nomenclatura caracterizadora de problemas de escalonamento, suficientemente abrangente, por cada uma das nomenclaturas relatadas ser insuficiente para classificar convenientemente muitos dos problemas que se pretendem tratar, académicos ou reais.

Neste trabalho foi desenvolvida uma nomenclatura para atingir este objectivo (Varela 1999, 2002a,b) baseado nos nomenclaturas disponíveis na literatura, anteriormente apresentados, de Conway e French (1967), Brucker (1995), Blazewicz (1996), Pinedo (2002) e Jordan (1996), assim como em informação adicional providenciada por, Graham et al (1979), Morton (1993), Artiba (1997) e Vollmann (1997).

Propõe-se uma nomenclatura, também do tipo $\alpha|\beta|\gamma$ que, à semelhança do que acontece com as nomenclaturas anteriormente apresentadas, inclui classes que se decompõem em subclasses que, por sua vez, incluem parâmetros, instanciáveis a vários níveis.

A classe α representa as características do ambiente de produção e decompõe-se nas subclasses $\alpha 1$ e $\alpha 2$.

A segunda classe permite especificar características, eventualmente inter-relacionadas, e restrições impostas no problema, relativas aos trabalhos e aos processadores e recursos auxiliares de produção, que são expressas pelos parâmetros β ($\beta_1, \ldots, \beta_{18}$).

A classe γ, à semelhança das nomenclaturas anteriormente apresentadas, expressa o critério de optimização, que pode incluir qualquer tipo de medida de desempenho simples ou complexa, nomeadamente medidas multi-critério.

Esta nomenclatura permite identificar as características subjacentes a cada problema de escalonamento e será usada como ponto de partida para a especificação de problemas baseada em XML, desenvolvida para representação e processamento de dados de escalonamento através de um sistema de web demonstrador que permita resolver estes problemas, através de um conjunto de métodos alternativos disponíveis através deste, numa comunidade distribuída de escalonamento.

Classe de parâmetros α

A primeira classe (α), como se referiu acima, é relativa aos parâmetros que permitem identificar o ambiente de produção e inclui duas subclasses α_1 e α_2. A subclasse α_1 é relativa à descrição do sistema de produção e inclui vários parâmetros para a identificação de diferentes tipos de sistemas de produção.

Os tipos de sistemas de produção baseiam-se nos ambientes propostos, apresentados nas secções 3.3.2, 3.3.3 e 3.3.4, que são: o sistema geral flexível de trabalhos multi-processador, o sistema geral flexível, o sistema geral de trabalhos multi-processador e o sistema geral.

Estes sistemas de produção podem ser de fase única ou de fases múltiplas. Os sistemas de fase única podem ser de processador único ou de processadores paralelos e estes últimos podem ainda decompor-se em sistemas de processadores paralelos idênticos, uniformes ou não relacionados. Os sistemas de fases múltiplas podem ainda decompor-se em sistemas do tipo oficina geral ou pura, linha geral ou pura ou sistema aberto e sistema aberto puro.

A subclasse α_2 consiste num único parâmetro, que permite definir a quantidade de processadores existentes no sistema.

O parâmetro $\alpha_1 \in$ {GFM, GF, GM, G} com:

GFM$\in$ {FM/f1, GFM/fm, FM/f1/P, FM/f1/PI, FM/f1/PU, FM/f1/PN, GFM/fm/J, FM/fm/JP, GFM/fm/F, FM/fm/FP, GFM/fm/O, FM/fm/OP}.

GF$\in$ {F/f1, GF/fm, F/f1/P, F/f1/PI, GF/f1/PU, F/f1/PN, GF/fm/J, F/fm/JP, GF/fm/F, F/fm/FP, GF/fm/O, F/fm/OP}.

GM$\in$ {M/f1, GM/fm, M/f1/P, M/f1/PI, M/f1/PU, M/f1/PN, GM/fm/J, M/fm/JP, GM/fm/F, M/fm/FP, GM/fm/O, M/fm/OP}.

G$\in$ {f1, G/fm, f1/P, f1/PI, f1/PU, f1/PN, G/fm/J, fm/JP, G/fm/F, fm/FP, G/fm/O, fm/OP}, que caracteriza o tipo de sistema, através de:

- α_1=GFM: sistema geral flexível de trabalhos multi-processador.
- α_1=FM/f1: sistema flexível de trabalhos multi-processador de fase única e processador único.
- α_1=GFM/fm: sistema geral flexível de trabalhos multi-processador de fases múltiplas;

- α_1=FM/f1/P: sistema flexível de trabalhos multi-processador de fase única e processadores paralelos.
- α_1=FM/f1/PI: sistema flexível de trabalhos multi-processador de fase única e processadores paralelos idênticos.
- α_1=FM/f1/PU: sistema flexível de trabalhos multi-processador de fase única e processadores paralelos uniformes.
- α_1=FM/f1/PN: sistema flexível de trabalhos multi-processador de fase única e processadores paralelos não relacionados.
- α_1=GFM/fm/J: sistema geral flexível de trabalhos multi-processador de múltiplas fases e ambiente de oficina.
- α_1=FM/fm/JP: sistema flexível de trabalhos multi-processador de múltiplas fases e ambiente de oficina pura.
- α_1=GFM/fm/F: sistema geral flexível de trabalhos multi-processador de múltiplas fases e ambiente de linha.
- α_1=FM/fm/FP: sistema flexível de trabalhos multi-processador de múltiplas fases e ambiente de linha pura.
- α_1=GFM/fm/O: sistema geral flexível de trabalhos multi-processador de múltiplas fases e ambiente de sistema aberto.
- α_1=FM/fm/OP: sistema flexível de trabalhos multi-processador de múltiplas fases e ambiente de sistema aberto puro.
- α_1=GF: sistema geral flexível.
- α_1=F/f1: sistema flexível de fase única e processador único.
- α_1=GF/fm: sistema geral flexível de fases múltiplas.
- α_1=F/f1/P: sistema flexível de fase única e processadores paralelos.
- α_1=F/f1/PI: sistema flexível de fase única e processadores paralelos idênticos.
- α_1=F/f1/PU: sistema flexível de fase única e processadores paralelos uniformes.
- α_1=F/f1/PN: sistema flexível de fase única e processadores paralelos não relacionados.
- α_1=GF/fm/J: sistema geral flexível de múltiplas fases e ambiente de oficina.
- α_1=F/fm/JP: sistema flexível de múltiplas fases e ambiente de oficina pura.
- α_1=GF/fm/F: sistema geral flexível de múltiplas fases e ambiente de linha.
- α_1=F/fm/FP: sistema flexível de múltiplas fases e ambiente de linha pura.
- α_1=GF/fm/O: sistema geral flexível de múltiplas fases e ambiente de sistema aberto.

- α_1=F/fm/OP: sistema flexível de múltiplas fases e ambiente de sistema aberto puro.
- α_1=GM: sistema geral de trabalhos multi-processador.
- α_1=M/f1: sistema de trabalhos multi-processador de fase única e processador único.
- α_1=GM/fm: sistema geral de trabalhos multi-processador de fases múltiplas.
- α_1=M/f1/P: sistema de trabalhos multi-processador de fase única e processadores paralelos.
- α_1=M/f1/PI: sistema de trabalhos multi-processador de fase única e processadores paralelos idênticos.
- α_1=M/f1/PU: sistema de trabalhos multi-processador de fase única e processadores paralelos uniformes.
- α_1=M/f1/PN: sistema de trabalhos multi-processador de fase única e processadores paralelos não relacionados.
- α_1=GM/fm/J: sistema geral de trabalhos multi-processador de múltiplas fases e ambiente de oficina.
- α_1=M/fm/JP: sistema de trabalhos multi-processador de múltiplas fases e ambiente de oficina pura.
- α_1=GM/fm/F: sistema geral de trabalhos multi-processador de múltiplas fases e ambiente de linha.
- α_1=M/fm/FP: sistema de trabalhos multi-processador de múltiplas fases e ambiente de linha pura.
- α_1=GM/fm/O: sistema geral de trabalhos multi-processador de múltiplas fases e ambiente de sistema aberto.
- α_1=M/fm/OP: sistema de trabalhos multi-processador de múltiplas fases e ambiente de sistema aberto puro.
- α_1=G: sistema geral.
- α_1=f1: sistema de fase única e processador único.
- α_1=G/fm: sistema geral de fases múltiplas.
- α_1=f1/P: sistema de fase única e processadores paralelos.
- α_1=f1/PI: sistema de fase única e processadores paralelos idênticos.
- α_1=f1/PU: sistema de fase única e processadores paralelos uniformes.
- α_1=f1/PN: sistema de fase única e processadores paralelos não relacionados.
- α_1=G/fm/J: sistema geral de múltiplas fases e ambiente de oficina.

- α_1=fm/JP: sistema de múltiplas fases e ambiente de oficina pura.
- α_1=G/fm/F: sistema geral de múltiplas fases e ambiente de linha.
- α_1=fm/FP: sistema de múltiplas fases e ambiente de linha pura.
- α_1=G/fm/O: sistema geral de múltiplas fases e ambiente de sistema aberto.
- α_1=fm/OP: sistema de múltiplas fases e ambiente de sistema aberto puro.

O parâmetro $\alpha_2 \in \{\varnothing, m\}$ refere-se à quantidade de processadores que integram o sistema:

- $\alpha_2=\varnothing$: assume-se que o número de processadores é variável.
- α_2=m: o número de processadores é igual a m (m é um valor inteiro positivo).

Se se tratar de um ambiente de processador único, por exemplo, um sistema geral flexível de trabalhos multi-processador, de fase única o sistema fica representado através da sigla GFM/f1. No caso de se tratar de um sistema multi-processador, por exemplo, um sistema geral flexível, de múltiplas fases, com ambiente de linha de produção pura, o sistema fica completamente identificado através da sigla GF/fm/FP.

Além disso, após se ter definido cuidadosamente o sistema de produção envolvido no problema convém averiguar quanto à quantidade de processadores que o constituem, visto que, este aspecto pode ser um dos principais indicadores do tipo de técnica que se deverá ou poderá aplicar em cada caso. Um exemplo disto é o caso das linhas puras com dois processadores (G/fm/FP2), que, como se sabe, remetem naturalmente, para o método de Johnson, de resolução rápida e eficaz!

A Tabela 4.10 resume estas características, em função das correspondentes definições apresentadas no capítulo 3.

Tabela 4.10 – Nomenclatura proposta: classe de parâmetros α.

Classe	Parâmetro	Designação	Valor	Valor defeito
α	α_1	Tipos de sistemas de produção: sistema geral flexível de trabalhos multi-processador, GFM: GFM de fase única ou GFM de múltiplas fases. GFM de fase única: GFM de processador único ou GFM de processadores paralelos. GFM de fase única e processadores paralelos: GFM de processadores paralelos idênticos ou GFM de processadores paralelos uniformes ou GFM de processadores paralelos não relacionados. GFM de múltiplas fases: GFM de oficina (ou oficina pura), GFM de linha (ou linha pura ou GFM de sistema aberto (ou sistema aberto puro). Sistema geral flexível, GF: GF de fase única ou GF de múltiplas fases. GF de fase única: GF de processador único ou GF de processadores paralelos. GF de fase única e processadores paralelos: GF de processadores paralelos idênticos ou GF de processadores paralelos uniformes ou GF de processadores paralelos não relacionados. GF de múltiplas fases: GF de oficina (ou oficina pura) ou GF de linha (ou linha pura) ou GF de sistema aberto (ou sistema aberto puro). Sistema geral de trabalhos multi-processador, GM: GM de fase única ou GF de múltiplas fases. GM de fase única: GM de processador único ou GM de processadores paralelos. GM de fase única e processadores paralelos: GM de processadores paralelos idênticos ou GM de processadores paralelos uniformes ou GM de processadores paralelos não relacionados. GM de múltiplas fases: GM de oficina (ou oficina pura) ou GM de linha (ou linha pura) ou GM de sistema aberto (ou sistema aberto puro). Sistema geral, G: G de fase única ou G de múltiplas fases. G de fase única: G de processador único ou G de processadores paralelos. G de fase única e processadores paralelos: G de processadores paralelos idênticos ou G de processadores paralelos uniformes ou G de processadores paralelos não relacionados. G de múltiplas fases: G de oficina (ou oficina pura) ou G de linha (ou linha pura) ou G de sistema aberto (ou sistema aberto puro).	GFM, GFM/f1, GFM/fm, GFM/f1/P, GFM/f1/PI, GFM/f1/PU, GFM/f1/PN, GFM/fm/J, GFM/fm/JP, GFM/fm/F, GFM/fm/FP, GFM/fm/O, GFM/fm/OP. GF, GF/f1, GF/fm, GF/f1/P, (PI, PU ou PN) GF/fm/J, GF/fm/JP, GF/fm/F, GF/fm/FP, GF/fm/O, GF/fm/OP. GM, GM/f1, GM/fm, GM/f1/P, (PI, PU ou PN) GM/fm/J, GM/fm/JP, GM/fm/F, GM/fm/FP, GM/fm/O, GM/fm/OP. G, G/f1, G/fm, G/f1/P, (PI, PU ou PN) G/fm/J, G/fm/JP, G/fm/F, G/fm/FP, G/fm/O, G/fm/OP.	1
	α_2	Quantidade de processadores no sistema	∅, variável, *m*	∅

O caracter nulo (∅) na coluna de valores dos parâmetros representa a característica do problema que se assume por defeito e, portanto, não figurará qualquer caracter na notação de classificação correspondente, relativo à característica em consideração. Assim, por exemplo, na Tabela 4.10, relativamente à quantidade de processadores, se se tiver especificado o sistema como sendo de processador único, não aparecerá nenhum caracter à frente da especificação do tipo de sistema de produção, para especificar o número de processadores, dado que é um processador apenas, o que, implicitamente, fica automaticamente expresso pelo caracter nulo "∅". Em termos gerais, poderá dizer-se que os diferentes tipos de sistemas equacionados neste trabalho visam cobrir diversos tipos de ambientes de produção que se possam encontrar na realidade industrial, cobrindo diversos tipos de sistemas, como se descreveu anteriormente, no capítulo 3, na secção 3.3.

Classe de parâmetros β

A classe relativa à caracterização dos lotes, trabalhos ou operações e dos processadores e recursos auxiliares é representada por β e contém uma série de parâmetros relativos a restrições ou condições de vária ordem, impostas no problema. Sendo assim, esta classe pode ser subdividida na subclasse dos parâmetros respeitantes aos trabalhos (β_1 a β_{11}) e na subclasses correspondentes aos processadores e recursos auxiliares (β_{12} a β_{18}).

A seguir apresentam-se as características relativas aos lotes, trabalhos ou operações. Exemplos de condições impostas a este nível são, restrições de precedência entre trabalhos, existência de prazos, para conclusão dos trabalhos e restrições nos tempos de processamento de trabalhos.

Parâmetros relativos aos trabalhos (β1 a β11)

$\beta_1 \in \{\varnothing$, pmtn, free-pmtn, comp-pmtn$\}$ refere-se à possibilidade de interrupção de trabalhos:

- $\beta_1 = \varnothing$: não existe interrupção.
- β_1 = pmtn: existe interrupção geral.
- β_1 = free-pmtn: existe interrupção livre.
- β1 = comp-pmtn: existe interrupção complexa.

$\beta_2 \in \{\varnothing$, prec, tree: intree, outtree, chain, sp-graph$\}$ define restrições de precedência dos trabalhos:

- $\beta_2 = \varnothing$: não existem restrições de precedência.
- β_2 = prec: existem restrições gerais de precedência.

- β_2 = tree (ou intree ou outtree): existem restrições de precedência em árvore.
- β_2 = chain: existem restrições de precedência em caminhos.
- β_2 = sp-graph: existem restrições de precedência em grafos.

$\beta_3 \in \{\varnothing, r_j, r_{ij}, r_{jl}, r_{ijl}\}$ define os instantes de chegada dos trabalhos ao sistema:

- $\beta_3 = \varnothing$: todos os trabalhos estão disponíveis para processamento num mesmo instante inicial (ex: zero, problema estático).
- $\beta_3 = r_j$: chegadas dinâmicas dos trabalhos ao sistema e descritas através de uma determinada função ou distribuição de chegadas (Ex: aleatória, Poisson, normal, exponencial negativa, etc.).
- $\beta_3 = r_{ij}$: chegadas dinâmicas dos trabalhos/ operações aos processadores do sistema e descritas através de uma determinada função ou distribuição de chegadas (Ex: aleatória, Poisson, normal, exponencial negativa, etc.).
- $\beta_3 = r_{jl}$: chegadas dinâmicas dos lotes/ famílias de trabalhos ao sistema e descritas através de uma determinada função ou distribuição de chegadas (Ex: aleatória, Poisson, normal, exponencial negativa, etc.).
- $\beta_3 = r_{ijl}$: chegadas dinâmicas dos lotes/ famílias dos trabalhos/ operações aos processadores do sistema e descritas através de uma determinada função ou distribuição de chegadas (Ex: aleatória, Poisson, normal, exponencial negativa, etc.).

$\beta_4 \in \{\varnothing, t_j=1, t_{ij}=1, t_j=p, t_{ij}=p, t_{i(j)} \in \{a,b\}, t_{min} \leq t_{i(j)} \leq t_{max}\}$ define a existência de restrições nos tempos de processamento:

- $\beta_4 = \varnothing$: os tempos de processamento são arbitrários.
- $\beta_4 = t_j=1$ ou $t_{ij}=1$: os tempos de processamento dos trabalhos/operações são unitários.
- $\beta_4 = t_j=p$ ou $t_{ij}=p$: os tempos de processamento dos trabalhos/operações são constantes.
- $\beta_4 = t_{i(j)} \in \{a,b\}$ ou $t_{min} \leq t_{i(j)} \leq t_{max}$: os tempos de processamento dos trabalhos/operações assumem valores de um intervalo ou conjunto de valores.

$\beta_5 \in \{\varnothing, d_l, d_j, d_{ij}, D\}$ define a existência de datas de entrega ou instantes de conclusão dos trabalhos:

- $\beta_5 = \varnothing$: não existem prazos para a entrega ou conclusão dos lotes/ trabalhos/ operações.
- $\beta_5 = d_k$ ou d_j ou d_{ij}: existem prazos para a entrega ou conclusão dos lotes/ trabalhos/ operações variáveis.
- $\beta_5 = D$: existe um prazo para a entrega ou conclusão dos lotes/ trabalhos/ operações fixo, que é igual para todos.

$\beta_6 \in \{\varnothing, \text{batch}\}$ refere-se à produção em lotes:

- $\beta_6 = \varnothing$: não existe produção em lotes.
- $\beta_6 = \text{batch}$: existe produção em lotes.

$\beta_7 \in \{\varnothing, \text{fam}\}$ é o parâmetro que define a existência de famílias de trabalhos:

- $\beta_7 = \varnothing$: não existe produção em famílias de trabalhos.
- $\beta_7 = \text{fam}$: existe produção em famílias de trabalhos.

$\beta_8 \in \{\varnothing, \text{compj}\}$ é o parâmetro que define a existência de trabalhos compostos:

- $\beta_8 = \varnothing$: não existe produção de trabalhos compostos.
- $\beta_8 = \text{compj}$: existe produção de trabalhos compostos.

$\beta_9 \in \{n, n_j, n_l)$ e representa a quantidade de trabalhos, operações ou lotes:

- $\beta_9 = n, n_j, n_l$: um número qualquer de trabalhos ou operações nos trabalhos ou trabalhos em lotes, respectivamente.
- $\beta_9 = n, n_j$ ou $n_l = k$ (2, 3, …): o número de trabalhos, operações ou lotes com um valor inteiro (positivo) especificado.

$\beta_{10} \in \{\varnothing, w_j, w_{ij}, w_{jl}\}$ define a existência de prioridades ou urgência dos trabalhos:

- $\beta_{10} = \varnothing$: não existem lotes/ trabalhos/ operações prioritários.
- $\beta_{10} = w_j$ ou w_{ji} ou w_{jl}: existem trabalhos, operações ou lotes de trabalhos prioritários.

$\beta_{11} \in \{\varnothing, mpt_j, mpt_{ij}, mpt_{jl}\}$ define a existência de trabalhos multi-processador:

- $\beta_{11} = \varnothing$: não existem trabalhos multi-processador.
- $\beta_{11} = mpt_j$ ou mpt_{ij} ou mpt_{jl}: existem trabalhos/ operações/ lotes multi-processador.

A Tabela 4.11 sintetiza a informação subjacente aos diferentes parâmetros de caracterização dos trabalhos.

Tabela 4.11 - Nomenclatura proposta: classe de parâmetros β dos trabalhos (β_1 a β_{11}).

Classe	Parâmetro	Designação	Valor	Valor por defeito
β	β_1	Interrupção de lotes/ trabalhos ou operações (tarefas)	∅, pmtn, free-pmtn, comp-pmtn	∅
	β_2	Restrições de precedência de lotes/ trabalhos ou operações	∅, prec, tree: intree, outtree, chain, sp-graph	∅
	β_3	Filosofia de chegadas de lotes/ trabalhos ou operações	∅, r_j, r_{ij}, r_{jl}, r_{ijl}	∅
	β_4	Tempo de processamento dos lotes/trabalhos/operações	∅, $t_j=1$, $t_{ij}=1$, $t_j=p$, $t_{ij}=p$, $t_{i(j)}\in\{a,b\}$, $t_{min}\leq t_{i(j)}\leq t_{max,\ldots}$	∅
	β_5	Prazos de entrega ou conclusão dos lotes/trabalhos/operações	∅, d_l, d_j, d_{ij}, D	∅
	β_6	Produção em lotes de trabalhos	∅, batch	∅
	β_7	Produção em famílias de trabalhos	∅, fam	∅
	β_8	Produção de trabalhos complexos	∅, compj	∅
	β_9	Quantidade de trabalhos, operações ou lotes a processar	n, n_j, n_l	n
	β_{10}	Prioridades ou urgência dos lotes/trabalhos/operações	∅, w_j, w_{ji}, w_{jl}	∅
	β_{11}	Lotes/ trabalhos/ operações multi-processador	∅, mpt_j, mpt_{ij}, mpt_{jl}	∅

A título de exemplo, na presença, por exemplo, de um parâmetro relativo à existência de restrições de precedência entre trabalhos (β_2) e sob escalonamento progressivo (*forward scheduling*), um trabalho pode ser iniciado apenas após todos os seus trabalhos predecessores estarem concluídos.

Os restantes parâmetros desta classe β (β_{12} a β_{18}) incluem parâmetros que indicam características no problema relacionados com os processadores e com os recursos auxiliares, tais como, existência de recursos críticos, disponibilidade dos processadores, ou outras restrições importantes como, por exemplo, impostas pela necessidade de uso de recursos auxiliares, tais como robôs e meios de transporte e/ou pela existência de armazéns locais, entre outros parâmetros, que são apresentados a seguir.

Parâmetros relativos aos processadores e aos recursos (β12 a β18)

$\beta_{12}\in\{\varnothing, eleg_k\}$ refere-se à elegibilidade dos processadores/ recursos:

- $\beta_{12}=\varnothing$: processadores/ recursos totalmente elegíveis.
- $\beta_{12}=eleg_k$: processadores/ recursos com elegibilidade condicionada, para os trabalhos.

$\beta_{13}\in\{\varnothing, avail_k\}$ é um parâmetro relativo à disponibilidade dos processadores/ recursos:

- $\beta_{13}=\varnothing$: processadores/ recursos sempre disponíveis.
- $\beta_{13}=avail_k$: processadores/ recursos com disponibilidade limitada.

$\beta_{14} \in \{\varnothing, auxk\}$ refere-se à existência de recursos auxiliares:

- $\beta_{14} = \varnothing$: não existem recursos auxiliares.
- $\beta_{14} = aux_k$: existem recursos auxiliares.

$\beta_{15} \in \{\varnothing, crtk\}$ refere-se à existência de recursos/ processadores críticos ou gargalos de estrangulamento no sistema:

- $\beta_{15} = \varnothing$: não existem processadores/ recursos críticos.
- $\beta_{15} = crt_k$: existem processadores/ recursos críticos (gargalos de estrangulamento).

$\beta_{16} \in \{\varnothing, s_{jk}, s_{ijk}, s_k\}$ define a existência de preparação nos processadores/ recursos:

- $\beta_{16} = \varnothing$: preparação dos processadores independente da ordem de lançamento dos trabalhos/ operações ou incluída nos tempos de processamento.
- $\beta_{16} = s_{jk}$: preparação dos processadores varia de acordo com a ordem de execução dos trabalhos;
- $\beta_{16} = s_{ijk}$: preparação dos processadores varia de acordo com a ordem de execução das operações dos trabalhos.
- $\beta_{16} = s_{lk}$: preparação dos processadores varia de acordo com a ordem de execução dos lotes de trabalhos.
- $\beta_{16} = s_k$: preparação externa dos processadores.

$\beta_{17} \in \{\varnothing, buffer_k, no\text{-}wait\}$ define a existência de armazéns locais:

- $\beta_{17} = \varnothing$: existem armazéns locais de capacidade ilimitada.
- $\beta_{17} = buffer_k$: existem armazéns locais (buffers) de capacidade limitada entre os estágios/ processadores.
- β_{17}=no-wait: não existem armazéns locais entre estágios ou associados aos recursos/ processadores.

$\beta_{18} \in \{\varnothing, mpm_k\}$ indica a existência de processadores multi-trabalho:

- $\beta_{18} = \varnothing$: não existem processadores multi-trabalho.
- $\beta_{18} = mpm_k$: existem processadores multi-trabalho.

A Tabela 4.12 resume as principais características dos parâmetros de classificação dos processadores e dos recursos auxiliares de produção subjacentes à nomenclatura de classificação proposta.

Tabela 4.12 - Nomenclatura proposta: classe de parâmetros β dos processadores (β_{12} a β_{18}).

Classe	Parâmetro	Designação	Valor	Valor por defeito
β	β_{12}	Elegibilidade dos recursos/ processadores	∅, M_k	∅
	β_{13}	Disponibilidade dos recursos ou processadores	∅, $avail_k$	∅
	β_{14}	Recursos adicionais/ auxiliares	∅, aux_k	∅
	β_{15}	Recursos/processadores críticos ou gargalos de estrangulamento no sistema	∅, crt_k	∅
	β_{16}	Preparação nos recursos/ processadores	∅, s_{jk}, s_{ijk}, s_{lk}, s_k	∅
	β_{17}	Armazéns locais ou buffers	∅, $buffer_k$, no-wait	∅
	β_{18}	Processadores flexíveis	∅, mpm_k	∅

Classe γ

Esta classe integra as características dos critérios de optimização do sistema de produção. Sendo assim, permite considerar no problema um vasto conjunto de objectivos a considerar, que incluem critérios regulares, não regulares e simples ou complexos.

No caso de se tratar de critérios regulares, estes podem ainda pertencer a diferentes subclasses ou tipos de critérios, como se explicou anteriormente na secção 3.4.

Exemplos de critérios de optimização que podem ser considerados, entre muitas outras, são:

$\gamma \in \{C_{max}, \Sigma C_j, \Sigma (w_j C_j), L_{max}, \Sigma T_j, \Sigma (w_j T_j), \Sigma E_j, \Sigma (w_j E_j), \Sigma N_{Tj}, \Sigma(w_j N_{Tj}),...(*)\}$

Com a inclusão desta última classe resulta então uma nomenclatura de classificação dos problemas de escalonamento com a seguinte estrutura: $\alpha_1, \alpha_2 \mid \beta_1, ..., \beta_{18} \mid \gamma$.

Como se referiu anteriormente, torna-se necessário recorrer a esta nomenclatura, um pouco mais expandida do que as apresentadas pelos restantes autores, de modo a ser possível efectuar uma representação razoavelmente completa e detalhada de um dado problema e abranger um amplo espectro de características dos problemas reais, recorrendo aos diferentes tipos de parâmetros apresentados.

Embora esta nomenclatura possa parecer bastante extensa, na prática, a classificação de um problema muito dificilmente incluirá todos os tipos de parâmetros referidos, até porque não existirão métodos capazes de resolver problemas com tamanha complexidade! Pelo que, na realidade, se manipularão estruturas classificativas de dimensão normalmente aceitável. A seguir ilustra-se o uso deste nomenclatura para classificar alguns problemas de escalonamento.

Exemplos de representação de classes de problemas:

Ex1: G/fm/F,3|n|Cmax, que se lê: programar um conjunto de n trabalhos independentes, com tempos de processamento arbitrários, disponíveis para serem realizados no instante inicial (por exemplo, no instante zero), a serem processados num sistema geral em linha, com três processadores, com o objectivo de minimizar o instante máximo de conclusão destes no sistema (*makespan* - Cmax).

Ex2: G/fm/F,3|n,d_j,$batch_j$|Cmax, que se lê da seguinte forma: programação de uma quantidade arbitrária de trabalhos, que têm prazos de entrega definidos, assim como tamanhos de lotes, a serem processados num sistema geral em linha (flow shop), com três processadores, no sentido de minimizar o instante máximo de conclusão dos trabalhos (C_{max}).

Dada a ausência de uma série de parâmetros na notação de representação do problema, isto significa que as características correspondentes ou estão ausentes do problema ou estão especificadas por defeito, correspondendo, ao símbolo ∅ que, portanto, não figurará na nomenclatura.

Muitos outros exemplos de aplicação desta nomenclatura podem ser apresentados, nomeadamente:

GFM/fm, m| n, r_j, pmtn, w_j, prec, t_j, d_j, availk |C_{max}.

G/f1/PI, m | n, w_j, tree, t_j, d_j, mpt_i, crt_s, prep | F_{med}.

GF/fm, m | $batch_l$, chain, t_j, mpt_i, bufferk | C_{max}.

G/fm/J, m | r_j, pmtn, prec, t_j, d_j | F_{med}.

GM/F1/PN, m | $batch_l$, n, pmtn, prec, t_j, d_j, mpt_i | w_jU_j.

G/fm/O, m | n, r_j, pmtn, w_j, graph, t_j, $batch_l$, d_j, mpt_i, bufferk, sjk | F_{med}.

G/fm/JP, 3 | n, prec, t_j, d_j, bufferk | F_{med}.

4.5 Considerações finais

No que se refere à caracterização da complexidade dos problemas, pode constatar-se que este aspecto não é, geralmente, um aspecto considerado importante, pelos diversos autores, em termos da sua inclusão numa nomenclatura de classificação dos problemas de escalonamento, embora existam bastantes autores a referir-se a este assunto, nomeadamente Blazewicz e Brucker, dada a importância deste. Este aspecto também não foi aqui

considerado importante para ser integrado na nomenclatura de classificação destes problemas, dado tratar-se de uma característica que, embora importante, tem uma relevância marginal no que concerne à problemática de associação de métodos para a resolução prática dos problemas.

Este aspecto poderá, porém, revelar-se de alguma importância quando, por exemplo, existem vários métodos alternativos, disponíveis para resolver um determinado problema, e dada a sua complexidade temporal (por exemplo, um problema do tipo *NP-complete*), poderá haver necessidade de seleccionar um método, de entre vários aplicáveis disponíveis, aquele ou aqueles que permitam encontrar uma solução, em tempo aceitável ou real. Sendo assim, este aspecto ou alguma característica relacionada, fará sentido ser abordada na classificação dos métodos de escalonamento. Muitas vezes este aspecto é bastante evidente, nomeadamente, quando se dispõe de métodos de classes muito diversas como, por exemplo, métodos optimizantes ou métodos de enumeração completa e métodos heurísticos baseados, por exemplo, em simples regras de sequenciação.

De igual modo, não se achou conveniente a inclusão de um parâmetro relativo à natureza das variáveis (determinística ou estocástica) na nomenclatura proposta, embora esta possa influenciar na selecção dos métodos adequados à resolução de um dado problema (Varela, 1999), uma vez que este aspecto também pode e deve apenas ser clarificado posteriormente, aquando da necessidade de resolver um problema, através de um determinado método. Assim, esta é, mais uma característica considerada no próprio método, em vez de ser considerada no problema de escalonamento.

Para concluir este capítulo, poderá dizer-se que, uma determinada nomenclatura será tanto mais adequada ao processo de classificação dos problemas quanto mais clara, rigorosa, precisa e objectiva esta for, por um lado, e por outro, quanto mais completa ou abrangente e detalhada for. Assim, um requisito fundamental será a utilização de uma nomenclatura classificativa o mais clara e objectiva possível que, além de ser compacta e precisa, seja também facilmente integrável e utilizada num processo de classificação de problemas permitindo, deste modo, o consequente processo de selecção de métodos para a sua resolução.

Pela comparação das nomenclaturas anteriormente apresentada pode constatar-se que não existe uma nomenclatura suficientemente abrangente, que permita representar a maioria dos problemas de escalonamento da produção que se podem encontrar na prática industrial. Deste modo, impõe-se a proposta de uma nova nomenclatura, que pode ser vista como uma

espécie de fusão entre nomenclaturas e ainda alargada para representar problemas de escalonamento que nenhuma outra permite representar.

Assim, neste trabalho propõe-se uma nomenclatura, estruturada em parâmetros, que tem em vista possibilitar uma classificação bastante exaustiva, capaz de se adequar à maioria dos problemas encontrados na teoria e prática de escalonamento. Contudo, esta nomenclatura, embora bastante exaustiva e abrangente podem existir problemas não representáveis.

5 Associação de Métodos a Problemas de Escalonamento da Produção

5.1 Introdução

A obtenção de bons programas de produção é bastante dependente dos métodos usados para resolver problemas de escalonamento. Por isso, há primeiro uma necessidade de identificar e claramente especificar os problemas e, depois, identificar métodos que os podem resolver para poder fazer escolhas apropriadas. Para alguns problemas podemos esperar encontrar vários métodos de resolução, assentes em técnicas diversas, como as referidas no capítulo 2, desenvolvidos por vários autores. Para outros, eventualmente apenas formas expeditas empíricas de resolução podem ser utilizadas, tais como simples regras de prioridade ou despacho.

O conhecimento da eficiência ou da complexidade dos métodos na resolução de problemas de escalonamento é importante para fazer a sua escolha e obter soluções rápidas, económicas, apropriadas e ajustadas ao problema a resolver.

Neste capítulo estrutura-se e organiza-se uma lista de métodos de escalonamento identificados na literatura e associando-os às classes de problemas que resolvem. Nesta associação, os métodos, depois de caracterizados, são identificados com a classe de problemas que resolvem sinteticamente representada através da nomenclatura desenvolvida neste trabalho e anteriormente apresentada no capítulo 4.

A associação de métodos a problemas faz-se usando o código do problema ou da classe de problemas, que é partilhado tanto pelos problemas como pelos métodos e definido na base da nomenclatura proposta. Assim, um método a que está associada uma classe de problemas, resolve todas as instâncias de problemas dessa classe. Cada instância é expressa, identificada ou especificada através de dados de entrada, estáticos e dinâmicos, relativos aos trabalhos e processadores necessários, assim como às restrições de disponibilidade e critério de optimização que o método equaciona. Todos estes dados resultam da parametrização de características ou atributos dos elementos envolvidos na classe de

problemas a resolver, como foram identificados nos capítulos 3 e 4, sendo de realçar os ambientes de produção, os trabalhos, os processadores e os recursos auxiliares e ainda os critérios de optimização.

Cada método é também identificado pelo seu nome de conhecimento público e/ ou pela referência bibliográfica da sua publicação.

Usando a terminologia e nomenclatura proposta nesta tese (capítulo 4, secção 4.4) começa-se por fazer a análise de casos mais simplificados evoluindo-se, gradualmente, para casos mais elaborados e gerais, na base, quer da disponibilidade de processadores alternativos, em cada fase de transformação, ou na necessidade de usar múltiplos processadores para a realização de uma operação de um dado trabalho.

5.2 Problemas de fase única e processador único

Os problemas de processador único (f1) têm sido objecto de estudo e alvo de investigação intensa desde os meados da década de 50, com trabalhos iniciais levados a cabo por Jackson e Smith (Brucker, 1995).

Neste ambiente o problema de escalonamento resume-se a um problema de ordenação dos trabalhos para produção no processador único disponível. Se esta ordenação se resolve antes do início da execução dos trabalhos, estamos perante um problema de sequenciamento; se se resolve durante o processo de execução, à medida que os trabalhos vão ficando concluídos e o processador disponível, diz-se que se resolve um problema de despacho – *dispatch problem*. Neste caso, geralmente os métodos resumem-se a regras simples de prioridade que, por estarem a resolver o problema de despacho, se designam por regras de despacho – *dispaching rules*. No entanto, as mesmas regras podem ser utilizadas para resolver o problema de sequenciamento. Neste caso o problema de sequenciamento é frequentemente resolvido em ambiente estático e por isso se refere normalmente como sendo um problema de escalonamento em ambiente estático.

Nesta secção apresenta-se uma amostra de métodos para resolver diferentes problemas em sistemas de produção de fase única com um único processador. A relevância destes problemas e dos métodos que os resolvem está no facto de se encontrarem casos práticos frequentes e ainda de poderem constituir base para a compreensão e resolução de problemas mais complexos, por exemplo através da resolução da classe f1 em ambientes de classe fm.

Tabela 5.1 – Métodos para problemas de processador único.

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação - Descrição sucinta**	**Referência**	**Complex.- qualidade**	**Tipo**	**Técnica**
1	f1||qualquer	Programar n trabalhos em processador único sujeita a um dado critério de optimização	Qualquer regra de prioridade	(revisão de regras)	Depende da regra	---	Regras
2	f1| | ΣCj	Programar n trabalhos em processador único minimizando o tempo médio de percurso	SPT- Shortest processing time rule	Baker (1974)	- Óptima	Optimizante	Regra
3	f1| | ΣAj	Programar n trabalhos em processador único minimizando o atraso médio	SPT- Shortest processing time rule	Baker (1974)	- Óptima	Optimizante	Regra
4	f1| | ΣAmax	Programar n trabalhos em processador único minimizando o atraso máximo	EDD- Earlist due date rule	Baker (1974)	- Óptima	Optimizante	Regra
5	f1| prec | Cmax	Programar n trabalhos em processador único minimizando o makespan	Algoritmo de Lenstra	Lenstra (1977)	O(n2)	---	---
6	f1| rj, pmtn, prec | Fmax	Programar n trabalhos com chegadas dinâmicas, interrupção e precedências em processador único minimizando o tempo de percurso máximo	Algoritmo de Baker et al.	Baker et al. (1983)	O(n2)	---	---
7	f1| rj, pmtn, tree | ΣCj	Programar n trabalhos com chegadas dinâmicas, interrupção e precedências (árvore) em processador único minimizando o tempo de percurso médio	Algoritmo de Lenstra & Rinnoy Kan	Lenstra & Rinnoy Kan (1980)	---	---	---
8	f1| fam | ΣCj	Programar famílias de trabalhos em processador único minimizando o tempo de percurso médio	Algoritmo de Coffman	Coffman (1990)	O(nlogn)	---	---
9	f1| fam, chains | ΣCj	Programar famílias de trabalhos com precedências (caminhos) em processador único minimizando o tempo de percurso médio	Algoritmo de Akers & Brucker (1990)	Brucker (1995)	---	---	---
10	f1| fam, prec, tj | ΣCj	Programar famílias de trabalhos com precedências em processador único minimizando o tempo de percurso médio	Algoritmo de Akers & Brucker	Brucker (1995)	O(n2)	---	---
11	f1|fam, rj, tree, tj | ΣwjCj	Programar famílias de trabalhos com chegadas dinâmicas e precedências (árvore) em processador único minimizando o tempo de percurso médio pesado	Algoritmo de Lenstra & Rinnoy Kan	Lenstra & Rinnoy Kan (1980)	---	---	---

5.3 Problemas de fase única e processadores paralelos

Apesar de aparentemente simples há vários casos de grande complexidade de problemas de fase única e processador único, como se mostrou na Tabela 5.1. Os problemas f1/P, numa perspectiva puramente empírica, podem ser considerados mais complexos do que os f1, por

envolverem mais processadores e, principalmente, por requererem a resolução de um problema adicional ao de sequenciamento, isto é, o problema de afectação.

Embora haja métodos susceptíveis de serem aplicados a esta classe genérica de problemas, é comum identificar métodos associados a classes mais refinadas, resultantes do uso de processadores paralelos idênticos, uniformes e não relacionados (f1/PI, f1/PU e f1/PN). As Tabelas 5.2 a 5.4 listam métodos para problemas diversos dentro destas classes.

Tabela 5.2 - Métodos para problemas de processadores paralelos idênticos.

	Problema		**Método**				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	f1/PI, m| pmtn | C_{max}	Programar trabalhos com interrupção em processadores paralelos idênticos minimizando o tempo de percurso máximo	Algoritmo de McNaughton	McNaughton (1959)	O(n) - Óptima	Optimizante	---
2	f1/PI, m | r_j, t_j, d_j | C_{max}	Programar trabalhos com chegadas dinâmicas, datas de entrega e restrições nos tempos de processamento em processadores paralelos idênticos minimizando o tempo de percurso máximo	Algoritmo de Simons	Simons (1983)	O(n3loglogn)	---	---
3	f1/PI, m| batchl, chains, tj, mpti, buffer | C_{max}	Programar lotes de trabalhos com precedências (caminhos) e operações multi-processador, e restrições nos tempos de processamento em processadores paralelos idênticos e buffers limitados minimizando o tempo de percurso máximo	Algoritmo de Simons	Simons (1983)	O(n3loglogn)	---	---
4	f1/PI, 3 | n | C_{max}	Programar n trabalhos em 3 processadores paralelos idênticos minimizando o tempo de percurso máximo	Algoritmo de Hoogeveen et al.	Hoogeveen et al. (1992)	---	---	---
5	f1/PI, m | r_j, t_j | C_{max}	Programar trabalhos com chegadas dinâmicas e restrições nos tempos de processamento em processadores paralelos idênticos minimizando o tempo de percurso máximo	Algoritmo de Simons	Simons (1983)	O(n3loglogn)	---	---
6	f1/PI, 2 | chains, t_j | C_{max}	Programar trabalhos com precedências (caminhos) e restrições nos tempos de processamento em 2 processadores paralelos idênticos minimizando o tempo de percurso máximo	Algoritmo de Hoogeveen et al.	Hoogeveen et al. (1992)	---	---	---

Tabela 5.3 - Métodos para problemas de processadores paralelos uniformes.

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	f1/PU| pmtn | C_{max}	Programar trabalhos com interrupção em processadores paralelos uniformes minimizando o tempo de percurso máximo	Algoritmo de McNaughton	McNaughton (1959)	O(n)	---	---
2	f1/PU, m | r_j, t_j, d_j | C_{max}	Programar trabalhos com chegadas dinâmicas, datas de entrega e restrições nos tempos de processamento em processadores paralelos uniformes minimizando o tempo de percurso máximo	Algoritmo de Simons	Simons (1983)	$O(n^3 \log\log n)$	---	---

Tabela 5.4 - Métodos para problemas de processadores paralelos não relacionados.

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	f1/PN| m | r_j, pmtn | L_{max}	Programar trabalhos com interrupção e chegadas dinâmicas em m máquinas paralelas não relacionadas minimizando o atraso máximo	Algoritmo de Programação Linear de Lawler & Labetoulle	Lawler & Labetoulle (1978)	---	---	Programação Linear
2	f1/PN, m| r_j, pmtn | ΣC_j	Programar trabalhos com interrupção e chegadas dinâmicas em m máquinas paralelas não relacionadas minimizando o tempo de percurso médio	Algoritmo de Bruno et al.	Bruno et al. (1974)	---	---	---
3	f1/PN, m | r_j, pmtn | ΣC_j	Programar trabalhos com interrupção e chegadas dinâmicas em m máquinas paralelas não relacionadas minimizando o tempo de percurso médio	Algoritmo de Horn	Horn (1973)	---	---	---
4	f1/PN, m | r_j, pmtn | L_{max}	Programar trabalhos com interrupção e chegadas dinâmicas em m processadores paralelos não relacionados minimizando o atraso máximo	Algoritmo de Programação Linear de Lawler & Labetoulle	Lawler & Labetoulle (1978)	---	---	Programação Linear

5.4 Problemas em sistemas de fases múltiplas de linhas de produção puras

Os ambientes em linhas de produção constituem um caso típico e frequente na indústria. Nesta secção apresentam-se exemplos de métodos que podem ser usados para resolver problemas de escalonamento que ocorrem em linhas de produção.

A seguir apresentam-se exemplos correspondentes a dois tipos de sistemas muito usuais, que são linhas puras (fm/FP) e linhas puras flexíveis (F/fm/FP).

Dentro da classe das linhas puras há ainda que distinguir entre linhas puras com dois, três ou m≥2 processadores, por se tratar de casos especiais, muito abordados na literatura, conforme se ilustra nas tabelas seguintes.

Tabela 5.5 – Métodos para problemas em linhas puras com 2 ou 3 processadores.

	Problema		**Método**				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	fm/FP, 2 | pmtn, prec | C_{max}	Programar trabalhos com interrupção e precedências numa linha pura com 2 processadores minimizando o makespan	Algoritmo de Muntz & Coffman	Muntz & Coffman (1970)	$O(n^2)$	---	---
2	fm/FP, 2 | pmtn, prec | C_{max}	Programar trabalhos numa linha pura com 2 processadores, minimizando o makespan	Regra de Johnson	Johnson (1954)	Polinomial	---	Regra
3	fm/FP, 2 | r_j, pmtn, prec | L_{max}	Programar trabalhos com interrupção, precedências e chegadas dinâmicas numa linha pura com 2 processadores minimizando o atraso máximo	Algoritmo de Lawler	Lawler (1982)	$O(n^6)$	---	---
4	fm/FP, 3 | | C_{max}	Programar trabalhos numa linha pura com 3 processadores minimizando o makespan	Algoritmo de Gonzales & Shahni	Gonzales & Shahni (1976)	---	---	---
5	fm/FP, 3| pmtn | ΣC_j	Programar trabalhos com interrupção numa linha pura com 3 processadores minimizando o tempo de percurso médio	Algoritmo de Liu & Bulfin	Liu & Bulfin (1985)	---	---	---

Outro tipo de problemas também bastante frequente é o problema de escalonamento em linhas de com mais do que dois processadores, como se ilustra na tabela seguinte.

Tabela 5.6 – Métodos para problemas em linhas puras com m processadores.

	Problema		**Método**				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	fm/FP, m | pmtn, tree | L_{max}	Programar trabalhos com interrupção e precedências (árvore) numa linha pura com m processadores minimizando o atraso máximo	Algoritmo de Garey & Johnson	Garey & Johnson (1976)	O(nmlognm)	---	---
2	fm/FP, m | pmtn, t_j, | $\Sigma w_i C_j$	Programar trabalhos com interrupção e restrições nos tempos de processamento numa linha pura com m processadores minimizando o tempo de percurso médio pesado	Algoritmo de McNaughton	McNaughton (Naughton, 1959)	O(nlogn)	---	---
3	fm/FP, m | t_j, d_j | ΣN_{Tj}	Programar trabalhos com datas de entrega e restrições nos tempos de processamento numa linha pura com m processadores minimizando o número de trabalhos atrasados	Algoritmo de Liu & Bulfin	Liu & Bulfin (1988)	O(nm+nlogn)	---	---

Outro tipo de problema bastante frequente é o de linhas flexíveis (F/fm/FP), que são linhas com processadores paralelos.

Neste tipo de sistema existe pelo menos uma fase ou estágio da linha constituída por dois ou mais processadores paralelos, tal como se referiu no capítulo .3

Deste modo, dependendo da natureza desses processadores paralelos, tratar-se-á, respectivamente, de linhas com processadores paralelos idênticos, uniformes ou não relacionados.

Na tabela seguinte apresentam-se alguns exemplos típicos de problemas que ocorrem em linhas deste tipo.

Tabela 5.7 – Métodos para problemas em linhas puras flexíveis.

	Problema		**Método**				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	F/fm/FP, m | n, prec, t_j, | C_{max}	Programar n trabalhos com restrições de precedência e nos tempos de processamento numa linha flexível pura com m processadores minimizando o tempo de percurso máximo	5 Algoritmos heurísticos (H1 a H5) (Gupta, 1996) e Lower bound and heuristics de Herman e Lee (1992)	Gupta (1997)	H1 a H4 (O(nlogn)) H5 (O(n3))	Heurístias	---
2	F/fm/FP, m | n, prec, t_j, bufferk, sjk | C_{max}	Programar n trabalhos com restrições de precedência e nos tempos de processamento numa linha flexível pura com m processadores com buffers limitados e setup minimizando o tempo de percurso máximo	Gilmore & Gomory Algorithms (1964)	Gilbert (1997)	---	---	---
3	F/fm/FP, m | n, agreg | C_{max}	Programar n trabalhos numa linha flexível pura com m processadores agregados minimizando o tempo de percurso máximo	Heurísticas e simulação de Morton	Morton (1993)	---	---	Simulação

5.5 Problemas em sistemas gerais flexíveis

Além dos sistemas de linhas de produção, anteriormente referidos e de acordo com a classificação dos sistemas de produção proposta neste trabalho, apresentada no capítulo 3 (secção 3.3) é referida um categoria genérica relativa a um ambiente de produção bastante relevante no mundo industrial, designado de sistema geral flexível (GF).

Sistemas deste tipo, além de contemplarem a possibilidade de existência de processadores paralelos alternativos nas diversas fases de produção, permitem incluir sistemas em que o número de operações de cada trabalho não é o mesmo. Genericamente, pode ser permitida a transposição de processadores e fluxo inverso e, ainda, entradas de trabalhos em pontos múltiplos do sistema e combinações destas características.

Dentro desta classe de sistemas podem, contudo, distinguir-se sistemas em que o fluxo é directo, i.e. unidireccional e uni-sentido, do tipo linha, como se exemplificou anteriormente. Na tabela seguinte apresentam-se alguns exemplos relativos aos problemas em ambiente geral flexível.

Tabela 5.8 – Métodos para problemas em sistemas gerais flexíveis.

	Problema		**Método**				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	GF/fm, m | n=k, r_j, prec | Σ_{wiTj}	Programar k trabalhos com chegadas dinâmicas restrições de precedência num sistema geral flexível com m processadores minimizando o atraso absoluto pesado	Algoritmo de Brucker	Brucker (1995)	---	---	---
2	GF/fm, m | n=3 | C_{max}	Programar 3 trabalhos num sistema geral flexível com m processadores minimizando o tempo de percurso máximo	Algoritmo de Brucker	Brucker (1995)	---	---	---
3	GF/fm, 2 | n, d_j, crt_s, aux_k | L_{max}	Programar n trabalhos com datas de entrega num sistema geral flexível com 2 processadores, processadores críticos e recursos auxiliares minimizando o atraso máximo	Heurísticas H1 e H2	Stevens (1997)	---	Heurística	---
4	GF/fm, m | n, batchl, prec, crt_s, agreg, setup | C_{max}	Programar n trabalhos em lotes com precedências num sistema geral flexível com m processadores, setup e processadores críticos e agregados minimizando o tempo de percurso máximo	Integer Optimization Formulation and a methodology based on a synergistic combination of LR, DP and heuristics	Luh (1997)	*NP-hard*	Heuristica	LR, DP

Tabela 5.8 - Métodos para problemas em sistemas gerais flexíveis (continuação).

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Classe**	**Descrição sucinta**	**Tipo**	**Classe**
5	GF/fm,m | rj, tj, dj | f:f∈{ Cmax, ΣCj	Programar trabalhos com chegadas dinâmicas, restrições nos tempos de processamento e datas de entrega num sistema geral flexível com m processadores minimizando o tempo de percurso máximo/ médio	Algoritmo de Brucker	Brucker (1995)	O(r(r3+rm+m 2,5) *mrnr+1)	---	---
6	GF/fm, m | n, $batch_l$, fam, r_j | FCFAM, MSFAM	Programar n trabalhos em lotes de famílias com chegadas dinâmicas num sistema geral flexível com m processadores minimizando o número de preparações e o tempo total de preparação	Group Scheduling Heuristics (GSH: FCFAM e MSFAM) + Simulação de Fmahmoodi & GEMartin	Fmahmoodi (1996)	---	---	---
7	GF/fm, m | n, prec, aux_k | C_{max}	Programar n trabalhos com precedências num sistema geral flexível com m processadores e recursos auxiliares minimizando o tempo de percurso máximo	Dispatch Heuristics	Morton (1993)	---	Heuristica	regra
8	GF/fm, m | n, w_j, d_j, bufferk| C_{max}	Programar n trabalhos com pesos e datas de entrega num sistema geral flexível com m processadores e buffers limitados minimizando o tempo de percurso máximo	Dispatch Methods – OPT-like heuristics e simulação	Morton (1993)	---	Heurística	OPT e simulação
9	GF/fm, m | n, $batch_l$, prec, crt_s, agreg, setup | C_{max}	Programar n trabalhos em lotes com precedências num sistema geral flexível com m processadores, processadores críticos e setup minimizando o tempo de percurso máximo	Integer Optimization Formulation and a methodology based on a synergistic combination of LR, DP and heuristics	Luh (1997)	---	Heurística	LR, DP
10	GF/fm, m | n, $batch_l$, fam, crt_s, aux_k | C_{max} + *balance workload*	Programar n trabalhos em lotes de famílias num sistema geral flexível com m processadores, processadores críticos e recursos auxiliares minimizando o tempo de percurso máximo e balanceando a carga de trabalho	Heurísticas para programação em sistemas com operadores Bhaskar	Vollmann (1997)	---	Heurística	---

Tabela 5.8 - Métodos para problemas em sistemas gerais flexíveis (continuação).

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Classe**	**Descrição sucinta**	**Tipo**	**Classe**
11	GF/fm, 5 | fam, r_j, d_j, buffer, sjk | Tempo de sjk + conteúdo total de trabalho + C_{max}	Programar trabalhos em famílias com chegadas dinâmicas e datas de entrega num sistema geral flexível com 5 processadores, setup e buffers limtitados minimizando o tempo de setup, o conteúdo total de trabalho e o tempo de percurso máximo	"Dynamic Scheduling Heuristics for dynamic manned manufacturing" (Regras de sequenciação: FCFS, SLACK, CDS, NEH)	Wem (1991)	---	Heurística	Regra
12	GF/f1/PI,2 |tj=1, chains| Cmax	Programar trabalhos com restrições nos tempos de processamento e precedências (caminhos) num sistema geral flexível com 2 processadores paralelos idênticos minimizando o tempo de percurso máximo	Hoogeveen et al.	Hoogeveen et al. (1992)	NP-hard	---	---
13	GF/fm/J, m| n=2| Cmax	Programar 2 trabalhos numa oficina geral flexível com m processadores minimizando o tempo de percurso máximo	Brucker & Schlie	Brucker & Schlie (1990)	O(nmax^3)	---	---
14	GF/fm/J, m| n=2, prec, rj | Lmax	Programar 2 trabalhos com precedências e chegadas dinâmicas numa oficina geral flexível com m processadores minimizando o tempo de percurso máximo	Jurisch	Jurisch (1993)	O(nmax^5)	---	---
15	GF/fm/O, m| tji=1, rj |f	Programar trabalhos com tempos de processamento unitários e chegadas dinâmicas num sistema aberto geral flexível com m processadores minimizando uma determinada função objectivo	Brucker et al.	Brucker et al. (1994)	O(r^2r!(mr!+ m^2,5) m^r*n^(r!r+1))	---	---
16	GF/fm/O, m| $\|C_{max}$	Programar trabalhos num sistema aberto geral flexível com m processadores minimizando o tempo de percurso máximo	Hoogeveen et al.	Hoogeveen et al. (1992)	NP-hard	----	---

5.6 Problemas em sistemas de fases múltiplas de sistemas abertos puros

Os problemas que ocorrem em sistemas gerais de fases múltiplas são caracterizados por um modelo multi-operação, isto é, associado a cada trabalho Jj existe um conjunto de operações Oj1, ..., Oj,nj. Nesta categoria de ambientes os processadores são dedicados, isto é, todos os conjuntos Mji são conjuntos de um único elemento. Além disso, existem restrições de precedência entre operações arbitrárias. Este é um modelo genérico, que reflecte um ambiente (sistema) de produção expresso por Brucker (1995) por G (alfa1=G). Os ambientes de oficinas de fabrico, de linhas e de sistemas abertos são casos particulares deste caso geral, tal como foi anteriormente referido no capítulo 3, na secção 3.3.

Os problemas que ocorrem em ambientes de sistemas abertos (fm/OP) constituem, portanto, um caso especial dos problemas em sistemas gerais em que, cada trabalho j consiste em m operações Oji (i=1,..., m), onde Oji tem de ser processada no processador Mi e não existem relações ou restrições de precedência entre operações.

Sendo assim, o problema consiste em encontrar sequências de trabalhos (operações pertencentes a um dado trabalho) e sequências de processadores (operações a serem processadas em cada processador). A Tabela seguinte ilustra alguns problemas deste tipo.

Tabela 5.9 – Métodos para problemas em sistemas abertos puros.

	Problema		**Método**				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	fm/OP, 2||Cmax	Programar trabalhos num sistema aberto puro com 2 processadores minimizando o tempo de percurso máximo	Gonzales & Sahni	Gonzales & Sahni (1976)	O(N)	---	---
2	fm/OP, m| rj, pmtn| Lmax	Programar trabalhos com chegadas dinâmicas e precedências num sistema aberto puro com m processadores minimizando o atraso máximo	Cho & Sahni	Cho & Sahni (1981)	---	---	---
3	fm/OP, 2| tji=1| ΣwjCj	Programar trabalhos com tempos de processamento unitários num sistema aberto puro com 2 processadores minimizando o tempo de percurso médio	Tanaev et al.	Tanaev e tal. (1989)	O(mn+nlogn)	---	---

Tabela 5.9 - Métodos para problemas de sistemas abertos puros (continuação).

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Classe**	**Descrição sucinta**	**Tipo**	**Classe**
4	fm/OP, m| tji=1, tree| Cmax	Programar trabalhos com tempos de processamento unitários e precedências (árvore) num sistema aberto puro com m processadores minimizando o tempo de percurso máximo	Braesel et al.	Braesel et al. (1991)	---	---	---
5	fm/OP, m| tji=1, rj| Lmax	Programar trabalhos com tempos de processamento unitários e chegadas dinâmicas num sistema aberto puro com m processadores minimizando o tempo de percurso médio	Labetoulle et al.	Labetoulle et al. (1984)	---	---	Prog. Linear
6	fm/OP, 2| tji=1, prec, rj| Lmax	Programar trabalhos com tempos de processamento unitários, precedências e chegadas dinâmicas num sistema aberto puro com 2 processadores minimizando o atraso máximo	Lawler	Lawler (1982)	O(n6)	---	---
7	fm/OP, m| tji=1| ΣUj	Programar trabalhos com tempos de processamento unitários num sistema aberto puro com m processadores minimizando o número médio de trabalhos atrasados	Liu & Bulfin	Liu & Bulfin (1988)	O(n2m)	---	---
8	fm/OP, m| tji=1| ΣTj	Programar trabalhos com tempos de processamento unitários num sistema aberto puro com m processadores minimizando o atraso absoluto médio	Liu & Bulfin	Liu & Bulfin (1988)	O(n2m)	---	---
9	Fm/OP, m| tji=1| ΣwjUj	Programar trabalhos com tempos de processamento unitários num sistema aberto puro com m processadores minimizando o número médio pesado de trabalhos atrasados	Brucker et al.	Brucker et al (1993)	O(n^2m^(m+1))	---	---

Tabela 5.9 - Métodos para problemas em sistemas abertos puros (continuação).

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Classe**	**Descrição sucinta**	**Tipo**	**Classe**
10	fm/OP, 2|| Lmax	Programar trabalhos num sistema aberto puro com 2 processadores minimizando o atraso máximo	Lawler et al.	Lawler et al. (1981)	NP-hard	Optimizante	---
11	fm/OP, 2| rj| Cmax	Programar trabalhos com chegadas dinâmicas num sistema aberto puro com 2 processadores minimizando o tempo de percurso máximo	Lawler et al.	Lawler et al. (1981)	NP-hard	Optimizante	---
12	fm/OP, 2|| ΣCj	Programar trabalhos num sistema aberto puro com 2 processadores minimizando o tempo de percurso médio	Achugbue & Chin	Achugbue & Chin (1982)	NP-hard	Optimizante	---
13	fm/OP, 2| tree| Cmax	Programar trabalhos com precedências (árvore) num sistema aberto puro com 2 processadores minimizando o tempo de percurso máximo	Lenstra	Lenstra	NP-hard	Optimizante	---
14	fm/OP, 3|| Cmax	Programar trabalhos num sistema aberto puro com 3 processadores minimizando o tempo de percurso máximo	Gonzales & Sahni	Gonzales & Sahni (1976)	NP-hard	Optimizante	---
15	fm/OP, 3| pmtn| ΣCj	Programar trabalhos com interrupção num sistema aberto puro com 3 processadores minimizando o tempo de percurso médio	Liu & Bulfin	Liu & Bulfin (1985)	NP-hard	Optimizante	---
16	fm/OP, 2| tji=1, prec| ΣUj	Programar trabalhos com tempos de processamento unitários e precedências num sistema aberto puro com 2 processadores minimizando o número médio de trabalhos atrasados	Brucker et al.	Brucker et al. (1993)	NP-hard	Optimizante	---

5.7 Problemas em sistemas de fases múltiplas de oficinas gerais

Os problemas que ocorrem em ambientes do tipo oficinas gerais (G/fm/J) são tipicamente problemas do tipo NP-completo, portanto, problemas geralmente difíceis, para os quais várias abordagens heurísticas têm vindo a ser desenvolvidas para os resolver. Tal como se apresentou no capítulo 2 (secção 2.4), os métodos de pesquisa local, tais como *simulated*

annealing (SA) , *tabu search* e algoritmos genéticos têm sido aplicados com sucesso a este tipo de problemas, permitindo obter bons resultados.

A tabela seguinte ilustra alguns problemas deste tipo.

Tabela 5.10 – Métodos para problemas em oficinas gerais.

	Problema		**Método**				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade-qualidade**	**Tipo**	**Técnica**
1	G/fm/J, 2| nj>=2| Cmax	Programar trabalhos com 2 ou mais operações numa oficina geral com 2 processadores minimizando o tempo de percurso máximo	Jackson	Jackson (1956)	O(nlogn)	---	---
2	G/fm/J, 2| tji=1| Cmax	Programar trabalhos com tempos de processamento unitários numa oficina geral com 2 processadores minimizando o tempo de percurso máximo	Timkowsky	Timkowsky (1985)	O(nlog(nr))	---	---
3	G/fm/J, 2| tji=1| Cmax	Programar trabalhos com tempos de processamento unitários numa oficina geral com 2 processadores minimizando o tempo de percurso máximo	Kubiak et al.	Kubiak et al. (1993)	O(nlog(nr))	---	---
4	G/fm/J, 2| tji=1| Lmax	Programar trabalhos com tempos de processamento unitários numa oficina geral com 2 processadores minimizando o atraso máximo	Timkowsky	Timkowsky (1994)	O(n^2)	---	---
5	G/fm/J, 2|| ΣCj	Programar trabalhos numa oficina geral com 2 processadores minimizando o tempo de percurso médio	Lenstra & Rinnooy Kan	Lenstra & Rinnooy Kan (1979)	NP-hard	---	---
6	G/fm/J, 2| tji={1,2}| Cmax	Programar trabalhos com restrições nos tempos de processamento numa oficina geral com 2 processadores minimizando o tempo de percurso máximo	Garey et al.	Garey et al. (1979)	NP-hard	---	---
7	G/fm/J, 2| tji=1, rj| ΣCj	Programar trabalhos com tempos de processamento unitários e chegadas dinâmicas numa oficina geral com 2 processadores minimizando o tempo de percurso médio	Timkowsky	Timkowsky (1994)	NP-hard	---	---
8	G/fm/J, 2| tji=1| ΣwjCj	Programar trabalhos com tempos de processamento unitários numa oficina geral com 2 processadores minimizando o tempo de percurso médio pesado	Timkowsky	Timkowsky (1994)	NP-hard	---	---

Tabela 5.10 - Métodos para problemas em oficinas gerais (continuação).

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Classe**	**Descrição sucinta**	**Tipo**	**Classe**
9	G/fm/J, 2| tji=1, rj| Cmax	Programar trabalhos com tempos de processamento unitários e chegadas dinâmicas numa oficina geral com 2 processadores minimizando o tempo de percurso máximo	Timkowsky	Timkowsky (1994)	O(n^2)	---	---
10	G/fm/J, 2| tji=1| ΣCj	Programar trabalhos com tempos de processamento unitários numa oficina geral com 2 processadores minimizando o tempo de percurso médio	Timkowsky	Timkowsky (1994)	O(nlogn)	---	---
11	G/fm/J, m| n=2| f	Programar 2 trabalhos numa oficina geral com m processadores minimizando uma determinada função objectivo	Brucker	Brucker (1988)	O(r^2logr)	---	---
12	G/fm/J, m| n=2, pmtn| f	Programar 2 trabalhos com interrupção numa oficina geral com m processadores minimizando uma determinada função objectivo	Sotskov	Sotskov (1991)	O(r^3)	---	---
13	G/fm/J, 2| n=k| ΣTj	Programar k trabalhos numa oficina geral com 2 processadores minimizando o atraso absoluto médio	Brucker	Brucker (1994)	O(r^3k)	---	---
14	G/fm/J, 3|| Cmax	Programar trabalhos numa oficina geral com 3 processadores minimizando o tempo de percurso máximo	Timkowsky	Timkowsky (1994)	NP-hard	---	---
15	G/fm/J, 3| tji=1| Cmax	Programar trabalhos com tempos de processamento unitários numa oficina geral com 3 processadores minimizando o tempo de percurso máximo	Lenstra & Rinnooy Kan	Lenstra & Rinnooy Kan (1979)	NP-hard	---	---
16	G/fm/J, 3| n=3| Cmax	Programar 3 trabalhos numa oficina geral com 3 processadores minimizando o tempo de percurso máximo	Sotskov & Shakhlevich	Sotskov & Shakhlevich (1993)	NP-hard	---	---

5.8 Problemas em sistemas gerais de fases múltiplas

Os problemas que ocorrem em sistemas gerais de fases múltiplas (G/fm) são muito usuais e a tabela seguinte apresenta uma amostra de problemas que ocorrem neste tipo de ambiente de produção.

Tabela 5.11 – Métodos para problemas em sistemas gerais de fases múltiplas.

	Problema		Método				
	Classe	**Descrição sucinta**	**Designação-descrição sucinta**	**Referência**	**Complexidade -qualidade**	**Tipo**	**Técnica**
1	G/fm, 2| n| Cmax	Programar n trabalhos num sistema geral de fases múltiplas com 2 processadores minimizando o tempo de percurso máximo	Strusevich	Strusevich (1991)	O(nlogn)	---	---
2	G/fm, 2| n, pmtn| Cmax	Programar n trabalhos com interrupção num sistema geral de fases múltiplas com 2 processadores minimizando o tempo de percurso máximo	Strusevich	Strusevich (1991)	O(nlogn)	---	---
3	G/fm, 2| n, pmtn| f	Programar n trabalhos com interrupção num sistema geral de fases múltiplas com 2 processadores minimizando uma determinada função objectivo	Shakhlevich & Sotskov	Shakhlevich & Sotskov (1994)	O(r)	---	---
4	G/fm, 2| tji={1,2}| C_{max}	Programar trabalhos com restrições nos tempos de processamento num sistema geral de fases múltiplas com 2 processadores minimizando o tempo de percurso máximo	Lenstra & Rinnooy Kan	Lenstra & Rinnooy Kan (1979)	NP-hard	---	---
5	G/fm, m| n=2| C_{max}	Programar 2 trabalhos num sistema geral de fases múltiplas com m processadores minimizando o tempo de percurso máximo	Shakhlevich & Sotskov	Shakhlevich & Sotskov (1994)	NP-hard	---	---
6	G/fm, m| n=2| ΣCj	Programar 2 trabalhos num sistema geral de fases múltiplas com m processadores minimizando o tempo de percurso médio	Shakhlevich & Sotskov	Shakhlevich & Sotskov (1994)	NP-hard	---	---

5.9 Considerações finais

Muitos outros exemplos poderiam ter sido apresentados, relativamente a cada um dos ambientes de produção mencionados, assim como de outros tipos, tais como sistemas relativos a outras situações de características "combinadas" e/ ou específicas, nomeadamente, problemas que ocorrem em sistemas gerais de trabalhos multiprocessador, nomeadamente, em sistemas de linhas e oficinas gerais de multiprocessador.

Algumas referências interessantes, onde mais exemplos podem ser encontrados, incluem Brucker (1995), Blazewicz (1996) e Jordan (1996).

Este modo de associação de métodos a problemas permite facilmente identificar, no espaço global de pesquisa, os métodos apropriados à resolução dos problemas de escalonamento, pelo que foi decidido adoptá-la como pressuposto fundamental ao desenvolvimento do sistema distribuído de apoio ao escalonamento industrial equacionado nesta tese e tratado no capítulo seguinte (sistema web de apoio ao escalonamento da produção - SWAEP).

6 Sistema Web de Apoio ao Escalonamento da Produção

6.1 Introdução

Este capítulo é central a este trabalho de investigação, visando contribuir para um melhor processo de resolução de problemas de escalonamento, através da proposta de uma arquitectura genérica de um sistema de web de apoio ao escalonamento da produção (SWAEP), baseado na utilização de métodos e conhecimento de escalonamento em geral, através de uma base de conhecimento distribuída, facilmente actualizável e acessível via Internet e intranets. Pretende-se aplicar este sistema em empresas industriais, possibilitando, assim, o desenvolvimento à medida de sistemas dedicados de escalonamento da produção com capacidade de expansão e partilha de conhecimento.

Para concretizar este objectivo é importante desenvolver um conjunto de instrumentos, i.e interfaces e mecanismos, necessários à implementação de um conjunto de funcionalidades requeridas para levar a cabo o processo de escalonamento. Estes instrumentos são usados nesta tese para testar a viabilidade do SWAEP através do desenvolvimento de um demonstrador. Com ele ilustram-se as funcionalidades principais do SWAEP na realização da actividade de escalonamento e desenvolvem-se alguns testes de aplicação.

O sistema deverá ser usado de forma interactiva para a resolução de problemas de escalonamento que ocorrem na indústria, com vista à satisfação de critérios de optimização diversos, em ambientes de produção industriais simples ou complexos, através da aplicação de métodos, local ou remotamente disponíveis e acessíveis via Internet e/ ou intranet, por empresas dos mais diversos sectores e operando em diversos contextos, nomeadamente comunidades, organizações ou empresas virtuais.

6.2 Arquitectura e funcionalidades do sistema

O conhecimento necessário à execução das trabalhos de escalonamento encontra-se, frequentemente, disperso por diferentes áreas de investigação e investigadores. Além disso, frequentemente as pessoas que pretendem resolver um problema não possuem conhecimento nem acerca de métodos existentes que podem ajudar na sua resolução nem da existência de formas de os classificar. Torna-se necessário conceber um sistema que possua uma arquitectura tal que permita a cooperação entre diferentes entidades, nomeadamente indivíduos ou empresas, de modo a facilitar o processo de resolução destes problemas, ou seja, um sistema capaz de, através de uma especificação do problema a resolver, rapidamente identifique métodos acessíveis para a sua resolução. Tais métodos podem estar disponíveis e acessíveis, quer local ou remotamente, nomeadamente através da Internet.

Um sistema desta natureza deve ser usado de forma interactiva na resolução dos problemas de escalonamento que ocorrem na indústria satisfazendo objectivos produtivos.

Nesta secção apresenta-se como solução uma arquitectura em ambiente multi-utilizador, baseado na web, em tecnologias cliente/ servidor e em XML e tecnologias associadas, que inclui uma base de conhecimento distribuída usada como repositório e partilha de conhecimento na área do escalonamento da produção industrial.

Funcionalidades do SWAEP

A arquitectura proposta resulta de um conjunto de funcionalidades requeridas pelo SWAEP. A hipótese de que a filosofia de escalonamento proposta tem viabilidade, exige que o SWAEP seja no mínimo capaz de realizar ou oferecer as seguintes funções e funcionalidades:

- Especificar e representar problemas de escalonamento.
- Especificar e univocamente representar métodos de escalonamento.
- Identificar métodos e sua localização na web, capazes de darem uma solução a um problema a resolver.
- Escolher o método de resolução.
- Especificar todas as variáveis necessárias e respectivos formatos para utilizar cada implementação informática de cada método, i.e. a sua assinatura.
- Ordenar a invocação de cada método para a obtenção de soluções.
- Instanciar as variáveis da assinatura do método.

- Especificar todas as variáveis ou medidas de resultados da utilização de um método na resolução de problemas.
- Tratar resultados de um problema.

Da análise das funções e funcionalidades expostas pode constatar-se que a resolução de problemas de escalonamento da produção envolve várias actividades, a começar pela clara definição de cada problema a resolver, através da sua identificação numa classe de problemas. Isto requer a utilização de um sistema de classificação e codificação de problemas.

A especificação e representação unívoca de métodos de escalonamento requer a classificação e codificação destes baseada no sistema de classificação e codificação de problemas.

A tarefa seguinte consiste em descobrir métodos adequados, se existirem, para a sua resolução. Esta tarefa requer uma função de identificação e localização na web de métodos para a resolução dos problemas e tem subjacente a necessidade de definir um esquema sistemático de associação de métodos a problemas de escalonamento e de poder obter, por exemplo, informação sobre o acesso para utilização do serviço de escalonamento associado a cada método.

A escolha do método de resolução será, naturalmente, dependente da utilidade do método e também da qualidade, eficiência e custo associado, quando houver métodos alternativos de escolha para um dado problema.

Uma vez escolhido um método torna-se, portanto, necessário proceder à especificação dos dados de entrada do métodos, i.e. das variáveis e respectivos formatos, para a utilização da correspondente implementação informática do método. Este processo consiste na especificação da assinatura do método e consiste numa função necessária para especificar cada instância de problema a resolver na forma que o método possa "entender" parar apresentar soluções.

A invocação de cada método, para a obtenção de soluções para os problemas é feita numa lógica de prestação de serviços web.

A instanciação das variáveis da assinatura do método é necessária para fornecer dados específicos de um problema a resolver, utilizando o método seleccionado.

A especificação das variáveis e medidas de resultados de um método, utilizado na resolução de um problema, é necessária para que o utilizador possa compreender e tratar as soluções fornecidas pelo método.

O tratamento de soluções dos problemas pode ser efectuado, por exemplo, apresentando-as num diagrama de Gantt ou tabela, ou inserindo-as num sistema ERP do utilizador para posterior tratamento e utilização.

Quando existem vários métodos alternativos para resolver um determinado problema, torna-se possível obter soluções alternativas, que devem ser avaliadas em função de critérios de optimização específicos que se pretendam considerar. Deste modo, torna-se possível melhorar o processo de resolução de um problema, através da comparação de diferentes soluções obtidas e, consequentemente, uma mais fácil tomada de decisão.

Num sistema desta natureza importa reunir problemas, métodos e resultados obtidos da execução de métodos implementados numa estrutura comum. Este sistema pode então ser usado para reportar aspectos teóricos e práticos associados a problemas de escalonamento da produção industrial, bem como a métodos existentes e futuros, que venham a ser disponibilizados em rede para a sua resolução. Assim, torna-se possível, não só aos implementadores de métodos darem a conhecer e disponibilizar os seus métodos para a resolução de problemas de escalonamento, como podem utilizadores usar o sistema para pesquisar problemas e métodos e, consequentemente, obter soluções para problemas que pretendam resolver.

Arquitectura do SWAEP

Aplicações distribuídas, caracterizadas pela execução de programas em diferentes processadores, permitem aliviar a sobrecarga destes, ao mesmo tempo que permitem a troca e a partilha de informação através de uma rede de comunicação e têm marcado uma adesão crescente no mundo da computação. Grande parte destas aplicações apresenta uma arquitectura do tipo cliente-servidor, em que o programa cliente solicita um serviço ao programa servidor que, após executá-lo, envia uma resposta ao cliente (Figura 6.1).

À semelhança de outros sistemas de web, este sistema segue uma arquitectura de três camadas lógicas que comunicam entre si. A camada de interacção com o cliente comunica com uma camada lógica intermédia que aloja a lógica da aplicação que, por seu lado, comunica com a camada terminal, que incorpora um repositório de informação e serviços. A camada de cliente permite aos utilizadores comunicarem com o sistema, para introdução

de dados e apresentação de resultados, através de browsers de web. A camada intermédia contém o núcleo do sistema, onde se encontram os servidores da aplicação, incluindo servidores de web e servidores de integração de XML. A camada terminal manipula repositórios de documentos XML e serviços de tratamento e comunicação de dados.

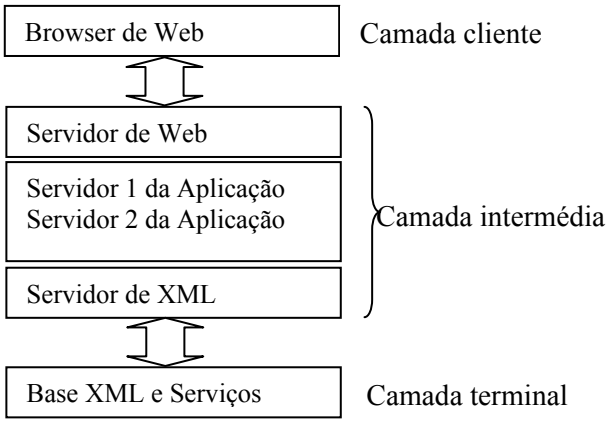

Figura 6.1 - Arquitectura de três camadas.

O sistema proposto é baseado em modelação XML e tecnologias associadas, incluindo um componente principal, que é uma interface para introdução, validação e transformação de dados de escalonamento da produção. A implementação desta interface no demonstrador do SWAEP é essencialmente controlada por documentos ASP (*active server pages*), DTD (*document type definition*) e XSL (*extensible markup language*) armazenados numa base de conhecimento distribuída, que inclui conhecimento de escalonamento da produção, guardado em documentos XML, que são validados de acordo com os correspondentes DTDs, antes de serem introduzidos nos correspondentes componentes de BC da BCD do sistema (Varela, 2002a; 2002b).

A interface é também responsável pelo mapeamento entre as características dos problemas definidos pelo utilizador e as especificações das entradas dos métodos, de modo a ser possível fornecer os dados do problema ao método que se pretende executar para o resolver, e no sentido inverso, uma vez obtidos os resultados da execução de um método, de modo a que estes sejam convertidos num determinado formato, para que possam ser apresentados ao utilizador. A Figura 6.2 ilustra esta arquitectura geral do sistema de escalonamento baseado em métodos local e remotamente disponíveis.

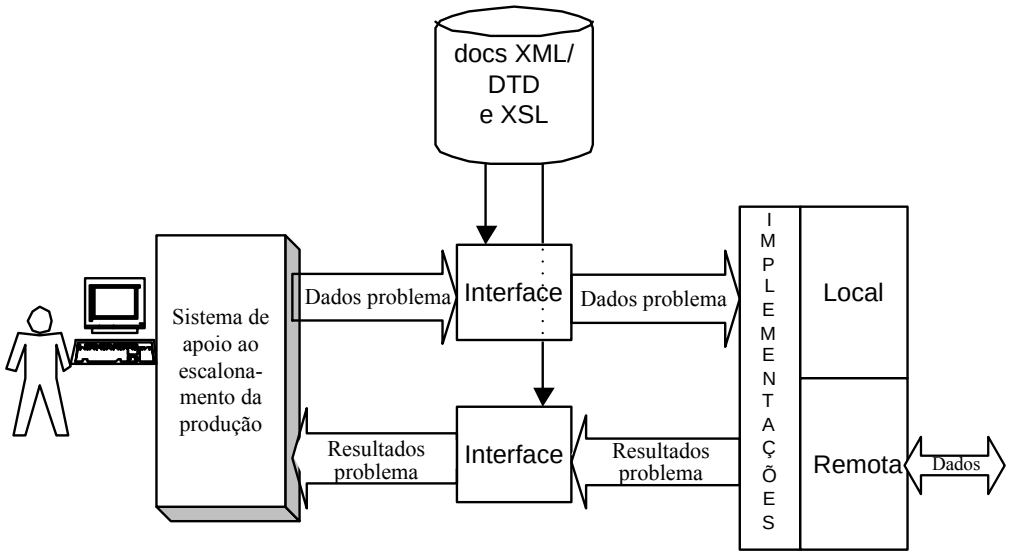

Figura 6.1 – Arquitectura geral do SWAEP.

O sistema visa providenciar apoio na tomada de decisão, através de uma selecção fácil e rápida de métodos adequados a cada situação ou problema particular, requerendo uma enquadramento e metodologia de selecção e execução de métodos que facilite e promova a partilha de conhecimento. As bases de conhecimento distribuídas (BCDs) constituem um cenário adequado para este objectivo de partilha de conhecimento relativo a problemas e métodos de escalonamento.

O SWAEP deve permitir a execução de métodos quer locais quer remotos. Os dados devem poder ser facilmente transferidos, lidos e processados, através do seu armazenamento e envio em rede no formato XML.

Base de conhecimento distribuída

As últimas tendências mostram, que os ambientes de computação são caracterizados por uma crescente heterogeneidade, distribuição e cooperação, onde as BCDs desempenham um papel de primordial importância (Wu, 1999).

Existem diversas formas de representação de conhecimento conhecidas e utilizadas ao longo das últimas décadas, no desenvolvimento de sistemas baseados em conhecimento, nomeadamente através de objectos, regras de produção, entre outras, tais como *frames*.

Neste trabalho o interesse recai, contudo, num tipo específico de base de conhecimento distribuída (BCD), que é uma base de conhecimento de web (Papazoglou, 2003). Nesta

BCD o conhecimento nos componentes de base de conhecimento está acessível via web, nomeadamente através da Internet.

O XML consiste numa sintaxe normalizada que permite a representação/ modelação de conhecimento e a comunicação no contexto deste tipo de BCD.

Estrutura ponto-a-ponto

Ambientes de rede, bases de conhecimento distribuídas e mecanismos inteligentes de pesquisa (*brokers*), para obtenção de informação a partir de servidores especializados e repositórios de conhecimento distribuídos através da Internet, permitem melhorar o processo de resolução de problemas industriais, nomeadamente de escalonamento da produção, através da partilha de conhecimento e de recursos.

Neste contexto, as tecnologias ponto a ponto (peer-to-peer, ou P2P) são emergentes e redes apropriadas adequam-se bem à crescente natureza descentralizada das companhias modernas e aos seus processos industriais e de negócio, quer se trate de uma empresa singular ou de um grupo de companhias em interacção (Papazoglou, 2003).

As redes P2P possuem potencialidades que permitem uma interacção directa entre os nós, o que torna o ambiente computacional descentralizado, nomeadamente em termos de armazenamento, processamento e troca de dados e de mensagens, bem como em termos de segurança e distribuição (Papazoglou, Terziyan, 2003).

O sistema aqui proposto segue um modelo de computação P2P, estando a BCD embebida numa rede desta natureza (http://www.oreilly.com/catalog/peertopeer/). Um conjunto de nós (*peers*), cada um contribuindo com a sua própria componente de base de conhecimento local, compõem uma base de conhecimento distribuída, formando uma rede com configuração P2P.

Neste tipo de rede cada nó possui, simultaneamente, características de cliente e de servidor, recebendo pedidos e enviando respostas, através da rede.

Um dos maiores benefícios deste tipo de rede, no contexto deste trabalho, é o de facilmente suportar o conceito de comunidade. Consequentemente, torna-se possível aos nós organizarem-se em grupos que podem colaborar entre si no intuito de atingir determinados objectivos. Um dos principais objectivos visados neste trabalho consiste em obter uma maior e melhor colaboração no processo de resolução de problemas de escalonamento da

produção, através da partilha de conhecimento e de métodos para a resolução destes problemas.

No sentido de satisfazer pedidos de utilizadores do sistema que permitam resolver problemas de escalonamento, cada nó da rede possui a sua própria base de conhecimento local, todas juntas formando uma base de conhecimento distribuída acerca de problemas e de métodos de escalonamento da produção.

O facto de cada nó da rede P2P possuir a sua própria base de conhecimentos local permite um melhor suporte à tomada de decisão, possibilitando aceder a um leque mais abrangente e diversificado de conhecimento de escalonamento da produção, bem como a diversas abordagens e métodos de resolução. Outra grande vantagem advém do facto de mais pessoas poderem estar envolvidas na manutenção e na actualização e enriquecimento do sistema.

Com base nesta arquitectura P2P, um utilizador pode ligar-se ao sistema a partir de um dos nós activos da rede de modo a descobrir métodos, na base de conhecimento local ou nas bases de conhecimento dos restantes nós, para a resolução de um determinado problema. Um método pode ser executado sob solicitação, podendo estar disponível através de um dos nós da rede ou mesmo num qualquer outro ponto externo à rede, desde que acessível através do sistema.

Os nós comunicam e colaboram entre si num processo de pesquisa de métodos de escalonamento adequados à resolução de problemas definidos pelo utilizador. Isto é possível através de um mecanismo que permita aos membros da rede P2P partilharem o seu conhecimento e disponibilizarem os métodos dentro e fora da organização, isto é, na rede P2P.

Este sistema baseia-se nos princípios de uma organização virtual (OV) (Camarinha-Matos, 1999; 2002). Nesta organização cada nó da BCD pode ser entendido como um membro/ sócio interessado em resolver problemas de escalonamento e, deste modo, à procura de soluções adequadas através da rede, que é então composta pela BC de cada membro participante. Sempre que um membro armazena conhecimento na sua BC local está, portanto, automaticamente a contribuir para o enriquecimento do repositório global de conhecimento (BCD).

Alguns nós da organização também podem actuar como nós especiais. Estes nós especiais dispõem da funcionalidade adicional de possuir a lista de todos os nós pertencentes à OV.

Tal lista contém informação acerca dos membros da OV e uma marca que indica o seu estado actual, que pode ser activo ou não activo. Um nó especial também permite configurar a rede P2P como um sistema aberto, permitindo que qualquer utilizador externo se junte à organização, ou como um sistema fechado, no sentido de que apenas nós pertencentes a uma determinada comunidade ou domínio possam integrar a organização. A Figura 6.3 apresenta um esboço geral da arquitectura P2P do sistema.

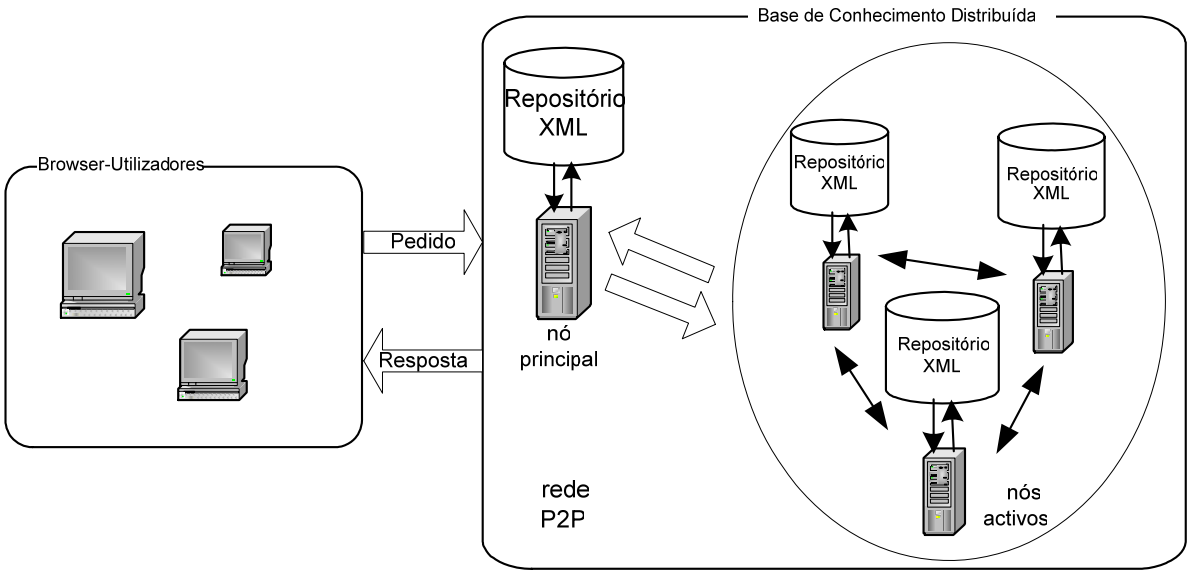

Figura 6.2 – Arquitectura ponto-a-ponto (P2P) do SWAEP.

Cada nó activo está constantemente à espera de pedidos de outros nós ou de utilizadores externos ligados através de browsers de web. Quando um pedido chega a um nó este começa por solicitar a lista dos restantes nós activos a um dos nós especiais ou principais. Este, de seguida, propaga o pedido a todos os nós activos dessa lista. Uma vez recebidas as contribuições dos restantes nós activos participantes no processo de satisfação do pedido (recolha de informação), os resultados são compilados e devolvidos ao utilizador que os solicitou.

A qualquer instante utilizadores externos podem juntar-se à OV e configurar-se como nós activos. Isto pode ser facilmente realizado, através da instalação de um conjunto de componentes comuns, que compõem a interface para aceder à rede e à sua BCD, uma unidade, existente em cada nó da OV.

Quando um novo nó se junta à OV envia um pedido com o seu endereço de IP aos nós especiais, que representam o domínio principal da rede P2P, que se tem de assegurar que esteja sempre disponível. Os nós especiais registam o endereço do novo nó, que será dinamicamente transferido aos restantes nós especiais da lista actual. Os membros da OV

podem juntar-se e permanecer ligados ou desconectar-se e abandonar a rede sempre que desejarem, o que configura características muito dinâmicas a este modelo de OV e rede P2P.

6.3 Demonstrador do SWAEP

Tal como se referiu anteriormente um SWAEP deve englobar várias funcionalidades, que incluem inserção de conhecimento acerca de problemas e métodos de escalonamento e correspondente pesquisa de informação. Os utilizadores devem puder fazer pedidos para visualização de classes de problemas e informação variada sobre métodos, incluindo classificação de métodos ou mesmo pesquisar informação sobre outros conteúdos/ conceitos apresentados pelo sistema. Os dados devem puder ser visualizados em diferentes formatos, usando diversas folhas de estilo (documentos XSL), apropriadas a cada tipo de pedido. Uma outra funcionalidade importante consiste na execução de métodos de escalonamento, dada a definição de um determinado problema. A selecção de um ou mais métodos específicos implementados é realizada através de um processo de pesquisa na BCD. O sistema permite diferentes formas de apresentação e armazenamento de resultados de modo a facilitar a comparação de diferentes soluções para um mesmo problema.

Estas funcionalidades do sistema derivam da estrutura de enquadramento e integração de conhecimento e de recursos e podem ser sumariadas como se segue:

- Classificação e identificação de problemas de escalonamento, usando uma notação específica, subjacente à nomenclatura proposta para o efeito.
- Inserção de conhecimento, acerca de problemas e métodos de escalonamento.
- Pesquisa conhecimento, acerca de problemas e métodos de escalonamento.
- Classificação e identificação de métodos de escalonamento.
- Associação de métodos a problemas de escalonamento, para a sua resolução.
- Execução de métodos diversificados para diferentes tipos de problemas de escalonamento e transformação de soluções em diferentes formatos.

A Figura 6.4 ilustra os principais processos subjacentes a estas funcionalidades do sistema, que são realizados por correspondentes módulos específicos.

Os processos são realizados por cinco módulos principais: um módulo de classificação de problemas e de métodos (MCPM), um módulo de inserção de conhecimento (MIC), um módulo de pesquisa de conhecimento (MPC), um módulo de invocação de métodos (MIM) e ainda um módulo de interface com o utilizador (MIU).

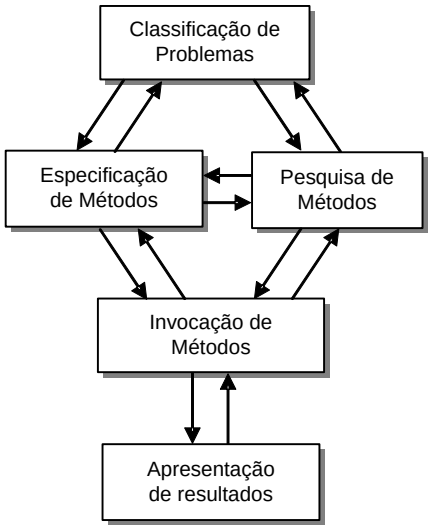

Figura 6.3 - Principais processos do sistema.

O MCPM permite identificar e classificar problemas de escalonamento, com base na nomenclatura de classificação de problemas proposta, previamente descrita na secção 4.4. Esta classificação de problemas constitui um processo inicial a partir do qual é possível aceder aos processos de inserção e pesquisa de novo conhecimento realizados através dos módulos MIC e MPC, respectivamente. Além disso, este módulo permite ainda identificar classes de métodos de escalonamento.

O MIC permite actualizar a BCD, repositório distribuído de conhecimento de escalonamento da produção, que inclui conhecimento relativo a problemas e a métodos. A inserção de conhecimento relativo a problemas requer a sua classificação prévia através da nomenclatura proposta e a de novos métodos requer a especificação da assinatura destes, que inclui a especificação das suas entradas e saídas e, adicionalmente, o endereço para a execução do método, bem como possível informação complementar sobre o método, para uma mais clara identificação e descrição deste, aquando da pesquisa de métodos adequados à resolução de problemas específicos.

Um outro módulo ou componente importante é o MPC, que permite a pesquisa de problemas e de métodos. A pesquisa de métodos integra um mecanismo de busca em *prolog*, para procurar métodos adequados à resolução de um determinado tipo de problema, na BCD. Além disso, os utilizadores podem fazer diversos tipos de pedidos de pesquisa, nomeadamente para visualização de informação diversa relativa a problemas e a métodos ou mesmo informação sobre outros conceitos relevantes apresentados pelo sistema.

A associação de problemas de escalonamento a métodos é efectuada usando informação disponível na BCD. O sistema é capaz de rapidamente associar métodos a problemas que ocorrem em ambientes de produção diversos.

O MIM é um módulo usado para aceder a métodos encontrados na BCD que, uma vez disponíveis, podem ser remotamente invocados para resolver problemas específicos colocados pelos utilizadores do sistema. Este módulo corre na forma de um serviço de web e é ilustrado mais adiante através do uso do protocolo de comunicação XML-RPC (*extensible markup language- remote procedure call*).

O módulo MIU é essencialmente controlado por documentos DTD e XSL, armazenados na BC local de cada nó pertencente à rede da organização virtual. Este módulo permite estabelecer toda a interacção necessária com o utilizador, nomeadamente introduzir dados relativos a especificações de problemas e de métodos e apresentar resultados em diferentes formatos, através de um browser de web.

O utilizador pode efectuar pesquisas diversas sobre problemas e métodos e fornecer informação específica ao sistema e receber respostas deste. Estas respostas poderão, por exemplo, aparecer em forma de notas explicativas, pedidos de mais informação e soluções para problemas, que podem ser visualizadas em diferentes formatos, através do uso de folhas de estilo (XSL) adequadas a cada tipo específico de pedido particular.

Com a modularização dos processos no sistema, suportada e enriquecida por uma arquitectura descentralizada do sistema do tipo P2P, torna-se possível obter maior qualidade na execução das funcionalidades disponibilizadas pelo sistema.

Nas secções seguintes faz-se uma descrição e ilustração das principais funcionalidades do demonstrador do sistema web de apoio ao escalonamento da produção (SWAEP). Estas funcionalidades incluem a especificação de problemas, métodos e resultados de escalonamento da produção. A pesquisa de problemas e métodos de escalonamento e a invocação de métodos para a resolução de problemas de escalonamento.

O demonstrador do SWAEP desenvolvido pode ser acedido através do seguinte endereço http://www.dps.uminho.pt.

6.3.1 Especificação de conhecimento de escalonamento da produção

O sistema demonstrador foi projectado em torno de dois conceitos principais: problemas e métodos de escalonamento da produção. No sentido de caracterizar problemas e de modo a

tornar esta tarefa mais fácil e claramente tratável recorreu-se a uma nomenclatura de classificação de problemas proposta (secção 4.4).

O sistema de web aqui apresentado permite a introdução de novo conhecimento relativo a problemas e a métodos de escalonamento da produção.

A BCD é fácil e dinamicamente actualizável através, não só, de peritos no domínio que pretendam especificar novo conhecimento no sistema, nomeadamente sobre novos métodos, mas também por outros utilizadores, que pretendam, por exemplo, resolver problemas de escalonamento e não encontrando informação sobre o problema que pretendam resolver podem acrescentá-lo à lista dos existentes. Isto é efectuado através do módulo de inserção de novo conhecimento (MIC), como foi previamente referido na secção 6.3.

Sendo assim, em cada nó da rede da OV, a correspondente BC pode ser constantemente actualizada e enriquecida com novas descrições de problemas e de métodos para a sua resolução, incluindo detalhes relativos a implementações destes métodos.

De modo a ser possível executar o serviço web para a invocação de métodos para a resolução de problemas de escalonamento, bem como outras funcionalidades do sistema, como a pesquisa prévia de métodos na BCD, torna-se então necessário introduzir previamente conhecimento relativo a problemas e a métodos no sistema, através da especificação de informação considerada relevante que, no caso dos métodos, inclui a definição das suas assinaturas e das suas localizações para posterior acesso, nomeadamente para ser possível obter informação relativa à associação de métodos a problemas.

A classificação dos problemas de escalonamento a resolver terá então de preceder a referida especificação dos métodos, o que é realizado com base na nomenclatura proposta (secção 6.3.1.1), de modo a ser possível uma associação entre problemas e métodos aquando da satisfação de pedidos de pesquisa e invocação destes métodos, fácil e parcialmente automática.

A linguagem XML foi considerada uma forma adequada de especificação dos conceitos de escalonamento da produção, que incluem dados complexos e estruturados, que devem poder ser facilmente acedidos e tratados através de um sistema facilmente acessível, utilizável, actualizável e expansível, como se ilustra mais adiante, ao longo deste capítulo, com a descrição e ilustração do sistema demonstrador desenvolvido neste trabalho (Varela, 2002a, 2002b). Optou-se por XML dadas as suas características e vantagens,

nomeadamente por se tratar de uma norma aberta, expansível, fácil de usar e aplicar a domínios concretos. Desta forma, conceitos de base de escalonamento da produção, como problemas e métodos, incluindo a especificação das suas implementações, em termos de entradas (dados) e saídas (resultados) foram modelados através de XML e correspondentes DTDs, tal como se ilustra na Figura 6.5.

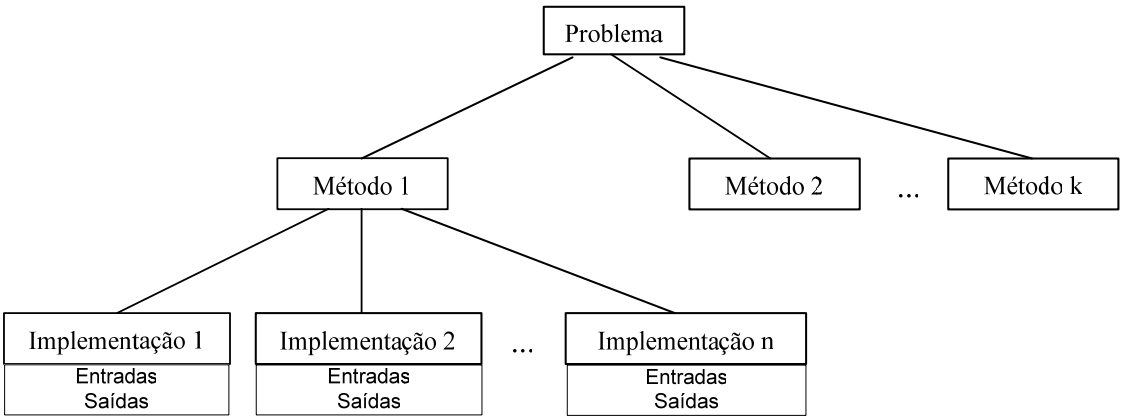

Figura 6.4 – Estruturação dos principais conceitos de escalonamento modelados.

6.3.1.1 Especificação de problemas

Para que seja possível especificar métodos para a resolução eficaz e eficiente de problemas de escalonamento torna-se necessária uma clara, objectiva e inequívoca caracterização de cada problema, através da especificação destes usando a nomenclatura de classificação previamente apresentada no capítulo 4.

Esta nomenclatura facilita o processo de associação de métodos a problemas, dado permitir sintetizar as principais características de cada problema, para a identificação destes, bem como servir de base à definição dos parâmetros de entrada de métodos para a sua resolução.

DTD de problemas

As características dos problemas são especificadas através de documentos DTD que contêm elementos que claramente os caracterizam. Trata-se de uma espécie de gramática que tem subjacente a nomenclatura de classificação proposta para os problemas de escalonamento da produção (Varela 2002a e 2002b). As classes ou tipos de problemas de escalonamento são então modelados por um documento DTD desenvolvido (Listagem 6.1). Este DTD é suficientemente genérico de modo a permitir definir diversificados problemas de escalonamento da produção.

```
<!-- Elements declaration -->
<!ELEMENT problems (problem+)>
```

```
<!ELEMENT problem (problemClass,problemComplexity?,context?,params?)>
<!ELEMENT params (alpha?, beta?, gamma?)>
<!-- Alpha elements -->
<!ELEMENT alpha (alpha1?, alpha2?)>
<!-- Alpha 1 -->
<!ELEMENT alpha1 EMPTY>
<!ATTLIST alpha1
system_type (0 | GFM | GF | GM | G | FM/f1 | FM/f1/P | FM/f1/PI | FM/f1/PU | FM/f1/PN | GFM/fm |
GFM/fm/J | FM/fm/JP | GFM/fm/F | FM/fm/FP | GFM/fm/O | FM/fm/OP | F/f1 | F/f1/P | F/f1/PI | F/f1/PU |
…) "0">
<!-- Alpha 2 -->
<!ELEMENT alpha2 EMPTY>
<!ATTLIST alpha2 machines (1 | 2 | 3 | m) "1">
<!-- Beta elements -->
<!ELEMENT beta (beta1?, beta2?, beta3?, beta4?, beta5?, beta6?, beta7?, beta8?, beta9?, beta10?, beta11?,
beta12? , beta13? , beta14? , beta15? , beta16? , beta17? , beta18?)>
<!-- Beta 1 -->
<!ELEMENT beta1 EMPTY>
<!ATTLIST beta1 preemption (0 | pmtn) "0">
<!-- Beta 2 -->
<!ELEMENT beta2 EMPTY>
<!ATTLIST beta2 precedences (0 | prec | tree | intree | outtree | chain | sp-graph) "0">
<!-- Beta 3 -->
<!ELEMENT beta3 EMPTY>
<!ATTLIST beta3 arrivals (0 | rj | rij) "0">
<!-- Beta 4 -->
<!ELEMENT beta4 EMPTY>
<!ATTLIST beta4 processing_times (1 | tj | tij | interval) "1">
<!-- Beta 5 -->
<!ELEMENT beta5 EMPTY>
<!ATTLIST beta5 deadlines (0 | dj | dij) "0">
<!-- Beta 6 -->
<!ELEMENT beta6 EMPTY>
<!ATTLIST beta6 batches (0 | batch) "0">
<!-- Beta 7 -->
<!ELEMENT beta7 EMPTY>
<!ATTLIST beta7 families (0 | fam) "0">
<!-- Beta 8 -->
<!ELEMENT beta8 EMPTY>
<!ATTLIST beta8 complex (0 | compj) "0">
<!-- Beta 9 -->
<!ELEMENT beta9 EMPTY>
<!ATTLIST beta9 jobs (0 | n | nj) "0">
<!-- Beta 10 -->
<!ELEMENT beta10 EMPTY>
<!ATTLIST beta10 priorities (0 | wj | wij) "0">
<!-- Beta 11 -->
<!ELEMENT beta11 EMPTY>
<!ATTLIST beta11 mpTask (0 | mpt) "0">
<!-- Beta 12 -->
<!ELEMENT beta12 EMPTY>
<!ATTLIST beta12 elegibility (0 | eleg) "0">
<!-- Beta 13 -->
<!ELEMENT beta13 EMPTY>
<!ATTLIST beta13 availability (0 | availk) "0">
<!-- Beta 14 -->
<!ELEMENT beta14 EMPTY>
<!ATTLIST beta14 auxiliary (0 | auxk) "0">
<!-- Beta 15 -->
<!ELEMENT beta15 EMPTY>
<!ATTLIST beta15 critical (0 | crtk) "0">
<!ELEMENT beta16 EMPTY>
<!ATTLIST beta16 setup (0 | stjk) "0">
```

```
<!-- Beta 17 -->
<!ELEMENT beta17 EMPTY>
<!ATTLIST beta17 buffers (0 | buffer) "1">
<!-- Beta 18 -->
<!ELEMENT beta18 EMPTY>
<!ATTLIST beta18 mpMachine (0 | mpm) "0">
<!-- Gamma element -->
<!ELEMENT gamma EMPTY>
<!ATTLIST gamma
mesure (Cmax | SumCi | SumWCi | Fmed | FWmed | Lmax | SumTi | SumWTi | SumUi | SumWUi | SumDi |
SumWDi | SumSi | SumWSi | SumEi | SumWEi) "Cmax">
```

Listagem 6.1 – Especificação DTD para problemas.

Este DTD permite definir o conteúdo possível de ser incorporado num documento XML contendo informação relativa a classes de problemas de escalonamento da produção e a forma como essa informação é estruturada e organizada e consequentemente aceite no correspondente documento XML (relativo a tipos/classes de problemas de escalonamento).

Da anterior Listagem 6.1 pode ler-se que, no intuito de definir um problema poderão, opcionalmente, definir-se parâmetros do tipo alfa, beta e gama, de acordo com a nomenclatura de classificação de problemas proposta (secção 4.4).

Os elementos do ficheiro DTD de problemas permitem caracterizar um problema de escalonamento, no sentido de que, do ponto de vista programático e de interacção com o sistema de apoio ao escalonamento da produção, um problema terá de ser descrito de acordo com esta gramática.

XML de problemas

Os documentos XML para a modelação de problemas incluem um conjunto de informação relativa à classificação de problemas de escalonamento da produção subjacente à nomenclatura proposta, apresentada na secção 4.4.

A Listagem 6.2 ilustra o documento XML relativo a classes de problemas, através de alguns exemplos típicos: f1|n|Cmax, f1/PI,m|pmtn,n|Cmax e fm/FP,2|n|Cmax.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--<?xml-stylesheet type="text/xsl" href="problems.xsl"?>
<!DOCTYPE problems SYSTEM "problems.dtd">-->
<problems>
        <problem>
                <problemClass>f1|n|Cmax</problemClass>
                <problemComplexity>Maximal Polinomially Solvable</problemComplexity>
                <context>Este problema é…</context>
                <params><alpha><alpha1 system_type=”1”/>
                        <alpha2 machines=”1”/><alpha>
                        <beta><beta9 jobs=”n”/></beta>
                        <gamma measure=”Cmax”/>
                </params>
        </problem>
```

```
    <problem>
        <problemClass>f1/PI,m|pmpt,n|Cmax</problemClass>
        <problemComplexity>Maximal Polinomially Solvable</problemComplexity>
        <context> Este problema é…</context>
        <params><alpha><alpha1 system_type="P"/>
            <alpha2 machines="m"/></alpha>
            <beta><beta1 preemption="pmpt"/>
            <beta9 jobs="n"/></beta>
            <gamma measure="Cmax"/>
        </params>
    </problem>
    <problem>
        <problemClass>fm/FP,2|n|Cmax</problemClass>
        <problemComplexity>Maximal Polinomially Solvable</problemComplexity>
        <context> Este problema é…</context>
        <params><alpha><alpha1 system_type="F"/>
            <alpha2 machines="2"/></alpha>
            <beta><beta9 jobs="n"/>
            <gamma measure="Cmax"/>
        </params>
    </problem>
    ...
</problems>
```

Listagem 6.2 – Código XML relativo a problemas.

Ilustração da especificação de problemas

A área de aplicação e o potencial do sistema de web são essencialmente dependentes da sua capacidade para especificar e lidar com problemas de escalonamento e a forma de os resolver. Isto depende essencialmente dos conteúdos da base de conhecimento distribuída (BCD) e, até certo ponto, também da capacidade de claramente especificar problemas no sistema.

Embora um considerável esforço tenha já sido distendido no desenvolvimento da nomenclatura proposta neste trabalho, capaz de permitir a especificação de uma grande variedade de problemas, poderá sempre ser expandida no sentido de alargar o conjunto de problemas que podem ser considerados pelo sistema, através da inclusão de novas características – parâmetros classificativos – dos problemas.

Considere-se a resolução do seguinte problema: produzir quatro trabalhos (n=4), que têm de ser processados numa linha de produção pura (*flow-shop*) com três processadores (m=3). Os tempos de processamento destes trabalhos (j) em cada um dos processadores (i) da linha são conhecidos e apresentados na Tabela 6.1 (tji).

Tabela 6.1 - Tempos de processamento da instancia de problema.

i / j	J1	J2	J3	J4
M1	3	11	7	10
M2	4	1	9	12
M3	10	5	13	2

O objectivo, conhecido à partida, consiste em minimizar o instante máximo de conclusão dos trabalhos no sistema (Cmax).

Para uma melhor ilustração das funcionalidades do sistema considere-se a necessidade de resolução da instância do problema fm/FP,m|n|Cmax, anteriormente descrita.

No sentido de proceder à classificação do problema, acabado de formular, será necessário identificar o conjunto de características descritas através de um conjunto de parâmetros divididos por três classes fundamentais, na forma $\alpha|\beta|\gamma$, relativos ao ambiente de produção (α), às características/ restrições dos trabalhos e dos processadores e recursos auxiliares de produção (β) e relativo(s) à critério de optimização do sistema (γ), respectivamente (secção 4.4), como se apresenta na Figura 6.6.

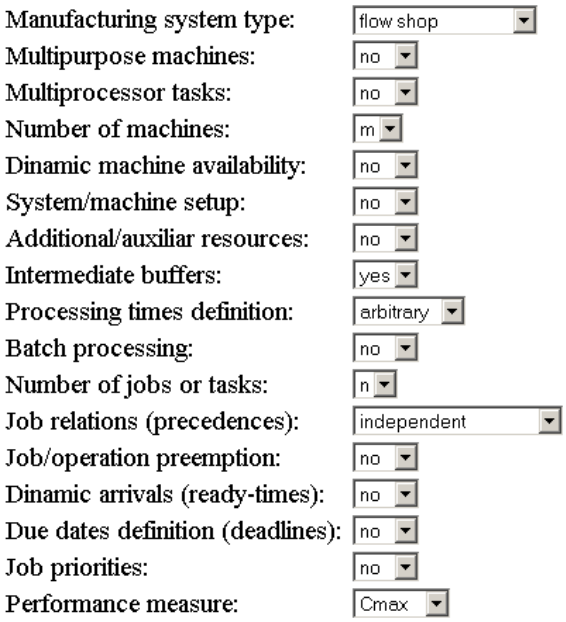

Figura 6.5 - Caracterização do problema.

Fazendo uma análise à Figura 6.6 pode resumir-se a seguinte especificação de parâmetros para o problema considerado:

Trata-se de um problema que ocorre numa linha de produção pura (*flow-shop*) (α1=fm/FP ou α1=F).

Não existem processadores multi-função neste sistema (*multipurpose machines*- mpm=0/no) e os trabalhos também não requerem processadores multi-item, que permitam processar vários trabalhos simultaneamente (*multiprocessor task*- mpt=0/no).

O número de processadores na linha é três (m=3) e estes encontram-se com disponibilidade ilimitada para processar os trabalhos (*dynamic machine availability*- avail=no).

Além disso, não se considera a questão da preparação dos processadores (em termos de tempo/custo, etc.) (*system/ machine setup*- setup=0/no) nem a existência de recursos adicionais ou auxiliares como, por exemplo, robôs ou transportadores (*additional/ auxiliary resources* – aux=0/no).

Os armazéns locais (buffers intermédios, entre processadores) no sistema possuem capacidade ilimitada (*intermediate buffers*- buffer=buffer/ yes).

Os tempos de processamento dos trabalhos nos processadores são arbitrários (a especificar aquando da resolução do problema, através de um método seleccionado, para a instância de problema considerada (*processing times definition*- tj=arbitrary) e não se equaciona o processamento em lotes (*batch processing*- batch=batch/ no).

O número de trabalhos a processar é genérico, a especificar aquando da resolução do problema (*number of jobs/ tasks*- n).

Estes trabalhos são independentes, isto é, não estão relacionados por restrições de precedência (*precedences*- prec=independent/ no).

Adicionalmente, não se considera a possibilidade de interromper os trabalhos, ou seja, uma vez iniciado o seu processamento num processador terá de ser finalizado nesse processador antes de passar para o estágio/ processador seguinte (*preemption*-pmtn=no).

Os trabalhos estão todos disponíveis para serem processados a partir de um mesmo instante inicial, por exemplo, no instante zero (*dynamic arrivals/ ready-times* – rj=0).

Não são conhecidos/ definidos prazos de entrega/ conclusão para os trabalhos (*due dates - deadlines* – dj=0) e estes também não têm atribuídas diferentes prioridades ou parâmetros de urgência (*priorities* – wj=0), entre outros parâmetros.

A critério de optimização consiste na minimização do instante máximo de conclusão dos trabalhos na linha (*performance measure* - Cmax).

Uma vez especificadas as características do problema, é possível definir o código da classe através da nomenclatura desenvolvida (secção 4.4). Neste caso o problema pertence à classe fm/FP,m|n|Cmax, como se ilustra na Figura 6.7.

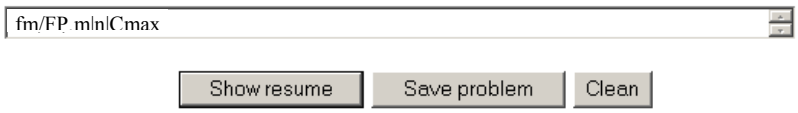

Figura 6.6 - Nomenclatura do problema.

Este código é importante na pesquisa de métodos de resolução.

Esta classe pode definir-se, de um modo muito resumido, da seguinte forma:

> *Processamento de trabalhos independentes numa linha de produção pura com três processadores, em ambiente estático, de modo a minimizar o tempo máximo de conclusão destes na linha.*

Naturalmente que, no caso de se terem submetido dados errados o sistema permite a sua correcção.

Uma vez especificado um novo problema este torna-se disponível para utilização futura, se necessário.

6.3.1.2 Especificação de métodos

Cada nó da BCD dispõe de um conjunto, mais ou menos alargado, de métodos de escalonamento. Deste modo, em muitos casos será possível e vantajoso considerar vários métodos alternativos, quando existem e estão disponíveis para a resolução de problemas de escalonamento da produção. Neste caso, diferentes resultados obtidos para um dado problema podem ser analisados. Soluções alternativas podem ser avaliadas e confrontadas e, desta forma, uma melhor tomada de decisão no escalonamento da produção poderá ser proporcionada, permitindo-se seleccionar a solução que se considere mais vantajosa.

Assinatura de métodos

Para que seja possível associar métodos a problemas de escalonamento da produção que se especifiquem torna-se necessário uma especificação prévia de cada problema e dos métodos correspondentes para a sua resolução. A especificação dos problemas é baseada na nomenclatura de classificação proposta (secção 0). A especificação dos métodos, quando disponibilizadas implementações destes para a resolução de instâncias de problemas requer a especificação das suas assinaturas.

A assinatura de um método inclui a definição dos seus parâmetros de entrada (*inputs*) e os seus parâmetros de saída (*outputs*). Além disso, torna-se ainda necessário especificar um nome/ endereço, através do qual o método possa ser invocado. Os elementos constituintes da assinatura de um método de escalonamento, à semelhança do que se passa com outros métodos, podem estar organizados de variadíssimas formas e ser expressos através de diversos tipos de elementos estruturais e combinações destes, nomeadamente, parâmetros, túplos, vectores e matrizes, tal como se ilustra na Figura 6.8.

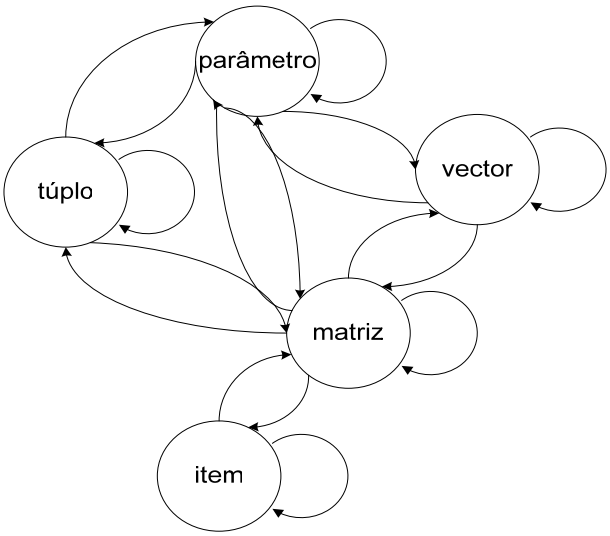

Figura 6.7 - Assinatura dos métodos.

Para um determinado método podem existir várias implementações disponíveis, que podem estar acessíveis, em rede, por exemplo através da Internet. Numa perspectiva programática, duas implementações de um determinado método podem diferir se, por exemplo, diferirem nas suas saídas. Nem todas as implementações funcionam do mesmo modo, sendo assim, torna-se necessário especificá-las no sistema, de modo que seja possível invocá-las para a resolução de instâncias de problemas. Esta especificação tem de incluir, o endereço URL (*uniform resource locator*) para invocar o método, eventualmente, a identificação do protocolo de comunicação e a assinatura do método implementado que, por seu turno, inclui a definição dos parâmetros que são necessários à sua invocação, i.e., as entradas, para fornecimento dos dados das instâncias de problemas a resolver e a especificação das suas saídas, i.e., os resultados obtidos para as instâncias dos problemas.

Os detalhes sobre os métodos e suas implementações são descritos por um DTD (secção 6.3.1.2 que permite validar a informação especificada antes de esta ser inserida no correspondente nó da BCD.

DTD de métodos

Os métodos de escalonamento e as respectivas implementações, à semelhança da descrição dos problemas, como atrás se referiu, também são descritas por documentos DTD. Quando os métodos têm implementações disponíveis, as suas especificações incluem a definição das suas assinaturas. A Listagem 6.3 ilustra o DTD relativo à especificação de métodos. As especificações DTD de métodos implementados são, subsequentemente, usadas para os invocar na forma de serviços (de web). Especificações de métodos não implementados incluem apenas informação, mais ou menos detalhada, sobre cada método, que pode ser acedida através de diferentes tipos de pesquisas, no repositório de documentos XML, relativos a problemas e a métodos de escalonamento da produção.

Muitos métodos podem ser adequados para resolver um dado problema, pelo que, num repositório XML de métodos deve ser possível pesquisar que métodos são adequados para resolver que problemas. Se um determinado método não tiver uma implementação disponível é possível efectuar pesquisas diversas, de modo a permitir, por exemplo, descobrir referências relativas a outros métodos, potencialmente adequados para resolver um dado problema.

Dado que diferentes implementações de métodos podem providenciar resultados de diferentes formas a descrição das suas saídas permite formatar os resultados que estes providenciam de diversos modos, como último passo do serviço de apoio à actividade de escalonamento da produção.

A informação relativa às assinaturas dos métodos, quando estes possuem implementações disponíveis, bem como informação adicional sobre estes, é expressa por um DTD da seguinte forma:

```
<!ELEMENT methods (method)*>
<!ELEMENT method(id,methodName,url?,problemCass,description?,methodClass?,methodSubclass?,
reference,complexity?,protocol?,notes?,signature?,gantt?)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT methodName (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT problemClass (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT methodClass (#PCDATA)>
<!ELEMENT methodSubclass (#PCDATA)>
<!ELEMENT reference (#PCDATA)>
<!ELEMENT complexity (#PCDATA)>
<!ELEMENT protocol (#PCDATA)>
<!ELEMENT signature (input,output)>
<!ELEMENT input (param | array | matrix)+>
<!ELEMENT param (#PCDATA)>
<!ATTLIST param name CDATA #REQUIRED type CDATA #REQUIRED control (submit) #IMPLIED>
<!ELEMENT array (item+)>
```

```
<!ATTLIST array from CDATA #FIXED "1" to CDATA #REQUIRED control (submit) #IMPLIED>
<!ELEMENT item (#PCDATA)>
<!ATTLIST item name CDATA #REQUIRED type CDATA #REQUIRED>
<!ELEMENT matrix (item+)>
<!ATTLIST matrix lines CDATA #REQUIRED columns CDATA #REQUIRED control (submit)
#IMPLIED>
<!ELEMENT output (param | array | matrix)+>
<!ELEMENT gantt (#PCDATA)>
```

Listagem 6.3 – Código DTD relativo à especificação de métodos.

XML de métodos

A informação correspondente aos métodos, incluindo a especificação das suas assinaturas, tem de ser validada, através do correspondente DTD, anteriormente expresso na Listagem 6.3, antes de ser introduzido no correspondente repositório XML da BCD.

A Listagem 6.4 apresenta uma amostra do documento XML relativo a métodos de escalonamento, através da ilustração de informação relativa à implementação do método de Johnson (1954), adequado para resolver problemas da classe fm/FP, 2| n| Cmax.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE methods SYSTEM "methods.dtd">
<methods><method>
                <id>32</id>
                <name>Johnson</name>
                <url>http://localhost:6002/RPC2</url>
                <problemClass>fm/FP,2|n|Cmax</problemClass>
                <description>Simple sequencing procedure…</ description>
                <methodClass>Sequencing Rule</methodClass>
                <methodSubclass>Simple Rule</methodSubclass>
                <reference>Johnson,1954</reference>
                <complexity>Maximal Polynomially Solvable</complexity>
                <protocol>XML-RPC</protocol>
                <signature>
                        <input>
                                <param name="n" type="integer"/>
                                <param name="m" type="integer" control="submit"/>
                                <matrix lines="m" columns="n">
                                        <item name="job" type="string"/>
                                        <item name="machine" type="string"/>
                                        <item name="t" type="double"/>
                                </matrix>
                        </input>
                        <output>
                                <param name="Cmax" type="double"/>
                                <param name="sequence" type="string"/>…
                                <matrix lines="m" columns="n">
                                        <item name="start" type="double"/>
                                        <item name="finish" type="double"/>…
                                </matrix>
                        </output>
                </signature>
                <gantt>yes</gantt>
        </method>…
</methods>
```

Listagem 6.4 – Código XML relativo a métodos: método de Johnson.

Tal como se pode constatar pelas listagens anteriores (6.3 e 6.4), existem dois níveis de informação que se tornam necessários especificar aquando da introdução de informação relativa a um novo método no sistema: informação genérica e informação detalhada.

A informação genérica do método, inclui o(s) autor(es), a referência, a(s) localização(ções) e, eventualmente, o protocolo usado na invocação do método, bem como outra informação facultativa, como a complexidade e uma descrição geral do método, a classe e a subclasse de métodos a que pertence, além de eventuais observações adicionais, por exemplo relativas à adequação do método para determinados tipos/ subtipos de problemas.

A informação mais detalhada do método é relativa à sua assinatura, que inclui informação relativa à especificação dos seus dados de entrada (*input*), necessários à sua execução e dados relativos aos seus resultados (*output*).

Na listagem anterior, e de acordo com o nomenclatura de classificação de problemas de escalonamento, as entradas incluem a definição do parâmetro n, inteiro, para o número de trabalhos a processar, um parâmetro m, inteiro, para o número de processadores e um conjunto de três itens, organizados numa matriz, que representam, respectivamente, a designação do trabalho, do tipo *string*, a designação do processador (*machine*), do tipo *string* e o tempo de processamento de cada trabalho em cada processador (tji), do tipo real. De modo similar, existe a definição das saídas (resultados) do método.

Após a especificação de um método ter sido introduzida num repositório XML de métodos tornar-se-á imediatamente acessível para pesquisas posteriores. Para o exemplo dado, após a inserção da definição do método de Johnson qualquer pesquisa de métodos compatível com a classe de problemas fm/FP, 2| n| Cmax incluirá este método nos resultados da pesquisa. Estas definições de métodos também permitem gerar automaticamente interfaces para a introdução de dados dos problemas, aquando da invocação de um método para a sua resolução e subsequentemente a apresentação de resultados, obtidos pela execução destes, tal como se ilustra mais adiante, na secção 6.3.3.

Ilustração da especificação de métodos

Uma vez efectuada a pesquisa de um método para resolver um problema, se não existir é possível usar o sistema para inserir informação sobre este, bem como disponibilizar a implementação do método, para que possa ser subsequentemente usado para resolver problemas colocados através do sistema. Suponha que se pretende inserir informação relativa ao método de Ignall-Schrage (Conway, 1967; Ignall, 1965), para a resolução de

problemas pertencentes à classe fm/FP,m|n|Cmax, uma classe que inclui a subclasse fm/FP,2|n|Cmax, anteriormente descrita na secção 6.3.1.1, para problemas de processamento de um qualquer número de trabalhos em linhas de produção puras (*flow-shop*) com m=2 processadores, consistindo neste caso numa generalização para qualquer número de processadores, m.

A Figura 6.9 ilustra a interface do sistema para a especificação deste método.

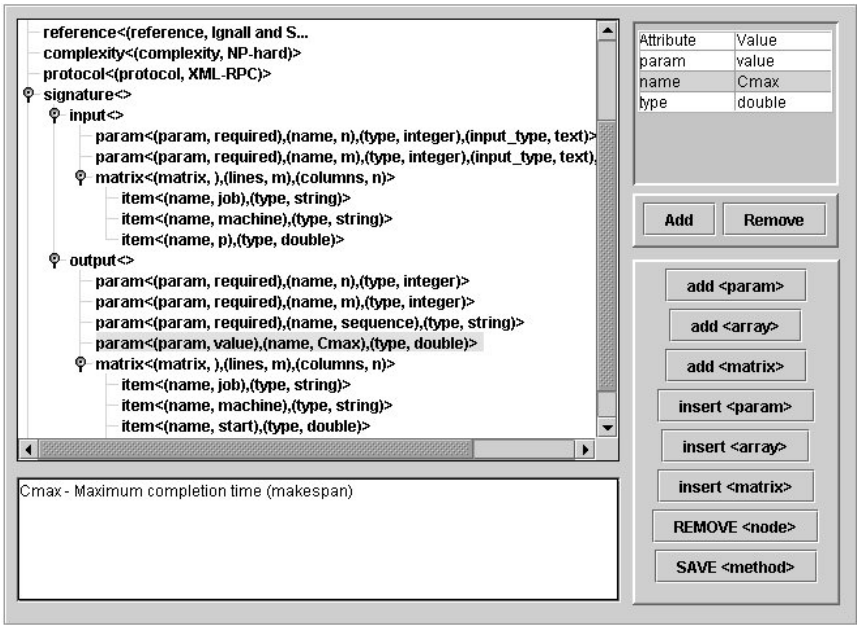

Figura 6.8 – Interface para especificação de novos métodos.

As entradas da assinatura, neste exemplo, incluem a definição de um parâmetro n, para o número de trabalhos a processar, do tipo inteiro, um parâmetro m, para o número de processadores no sistema (linha) e ainda um conjunto de três itens numa estrutura matricial, de dimensão m linhas por n colunas, que representam, respectivamente, o designação dos trabalhos (*job*), a designação dos processadores (*machine*), ambos do tipo *string* e o tempo de processamento de cada trabalho em cada processador (t, do tipo *double* - real). De um modo similar, existe também a definição das saídas da implementação do método. Ou seja, os parâmetros n, para o número de trabalhos e m, para o número de processadores da linha, que aparecem também como dados de saída do método e, mais uma vez, uma matriz, com os itens *job* e *machine*, relativos às designações dos trabalhos e dos processadores, respectivamente e, adicionalmente, os intens *start* e *finish*, do tipo real (*double*), relativos aos tempos de início e de fim de cada trabalho em cada processador da linha.

Existe ainda um outro resultado dado por este método, que é o instante máximo de conclusão dos trabalhos no sistema, Cmax, que também é do tipo real (*double*) e a

sequência de fabrico dos trabalhos na linha, sequência que, portanto, neste tipo de problemas, em linhas puras – *flowshop* - se mantém para todos os processadores da linha.

Esta informação, uma vez gravada, será subsequentemente inserida no documento XML de métodos, um vez validada contra o documento DTD associado, tal como previamente ilustrado na secção 6.3.1.2, de modo a ser incorporada na correspondente BC local do nó da rede OV da BCD.

Para o exemplo apresentado, após confirmação de gravação a informação será inserida no correspondente componente de BC local da rede da OV. Estes dados, relativos à definição do método, integrarão então o correspondente documento XML relativo a métodos, uma vez validados contra o documento DTD associado (Listagens 6.3 e 6.4). Posteriormente, ficarão disponíveis para processos de pesquisa que incluam o método em questão, para a resolução de problemas pertencentes à classe bem como a subclasses desta classe, nomeadamente, a classe fm/FP,2|n|Cmax. Além disso, solicitações para invocação do método passarão também a permitir gerar automaticamente interfaces para a entrada de dados dos problemas e apresentação dos correspondentes resultados através de interfaces também geradas automaticamente pelo sistema.

A disponibilização de novos métodos permite o enriquecimento da BCD, a partir da actualização de cada BC local. Este processo de introdução de novo conhecimento torna os processos de selecção e de uso de métodos de escalonamento muito mais ricos, para a resolução de uma ampla e variada gama de problemas de escalonamento que possam ocorrer no mundo industrial.

Deste modo, é nossa convicção que, diversos tipos de métodos, quer heurísticos quer analíticos podem e devem ser disponibilizados, de uma forma fácil, através de um sistema desta natureza.

6.3.1.3 Especificação de resultados

A especificação de resultados ou saídas dos métodos é igualmente importante, na medida em que permite que diversas alternativas de apresentação de resultados possam ser usadas para as soluções dos problemas colocados através do sistema.

Os diagramas de Gantt constituem uma forma muito típica e útil de apresentação de resultados que podem ser facilmente gerados, de uma forma automática, a partir da formatação das respostas (saídas) providenciadas pela invocação de métodos. Isto é conseguido graças à modelação das saídas ou respostas dos métodos através de um

documento XML e correspondente DTD. Tal modelação permite uma forma fácil de conversão de saídas em diferentes formas de apresentação dos resultados ao utilizador do sistema, nomeadamente através de diagramas de Gantt e tabelas diversas.

DTD de resultados

A Listagem 6.5 ilustra a especificação DTD relativa a resultados de problemas, expressos através de um diagrama de Gantt.

```
<!ELEMENT gantt (problem*)>
<!ELEMENT problem (job+)>
<!ATTLIST problem
        id CDATA #REQUIRED
        class CDATA #REQUIRED>
<!ELEMENT job (machine+)>
<!ATTLIST job
        id CDATA #REQUIRED
        name CDATA #REQUIRED>
<!ELEMENT machine (start+,finish+)>
<!ATTLIST machine
        id CDATA #REQUIRED
        name CDATA #REQUIRED>
<!ELEMENT start (#PCDATA)>
<!ELEMENT finish (#PCDATA)>
```

Listagem 6.5 – Código DTD relativo a diagramas de Gantt.

XML de resultados

A Listagem 6.6 exemplifica código XML, ilustrando os resultados para um problema fm/FP,3|n|Cmax, obtidos pela aplicação do método de Ignall-Schrage, para problemas fm/FP,m|n|Cmax, a serem expressos através de um diagrama de Gantt.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gantt SYSTEM "gantt.dtd">
<gantt>
   <problem id="0010" class=" fm/FP,3|n|Cmax">
        <job id="J1" name="Job1">
          <machine id="001" name="M1"><start>0</start><finish>3</finish>…</machine>…
        </job>…
   </problem>
</gantt>
```

Listagem 6.6 – Código XML relativo a diagramas de Gantt.

Ilustração da especificação de resultados

A Figura 6.10 ilustra um diagrama de Gantt que representa a solução obtida para o problema considerado, com um instante de conclusão de 39 unidades de tempo.

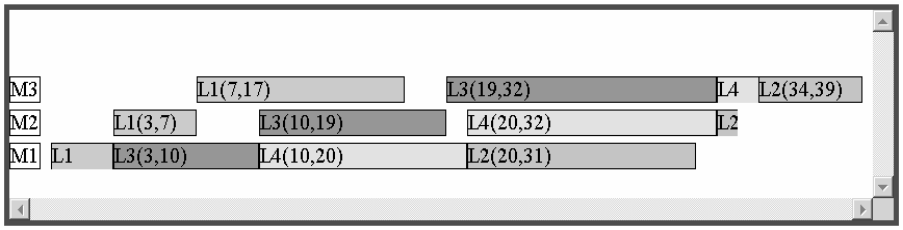

Figura 6.10 – Expressão de resultados através de um diagrama de Gantt.

Os diagramas de Gantt continuam a ser muito utilizados devido ao seu elevado poder expressivo, permitindo uma forma relativamente fácil de análise e comparação de resultados obtidos pela execução de diferentes métodos para um mesmo problema.

Existem muitas outras alternativas para apresentação dos resultados dos problemas, nomeadamente a apresentação directa de resultados através de documentos XML ou qualquer outra formatação de resultados, como por exemplo, em tabelas com formatações diversas, como se ilustrará mais adiante na secção 0, aquando da ilustração da invocação de métodos para a resolução de problemas.

6.3.2 Pesquisa na base de conhecimento distribuída

Uma das funcionalidades importantes deste sistema consiste em pesquisar conhecimento de escalonamento da produção. Existem várias formas de pesquisa disponíveis, nomeadamente pesquisa por classes de problemas e de métodos. Outra forma de pesquisa consiste em encontrar métodos que satisfaçam os requisitos especificados pelo utilizador para um determinado problema de escalonamento colocado a resolver. Sempre que as implementações dos métodos estejam disponíveis, estas podem ser usadas para resolver os problemas. Este conhecimento relativo a problemas e métodos encontra-se disperso pela BCD e acessível na rede.

Para levar a cabo a execução de métodos torna-se necessário realizar alguns passos prévios incluindo acesso a conhecimento intermédio acerca de métodos e suas implementações para a resolução dos problemas. Sendo assim, embora algum conhecimento disponibilizado acerca de métodos seja estritamente desnecessário, do ponto de vista de acesso a métodos para a resolução de problemas, achou-se conveniente a sua inclusão, no sentido de ser possível disponibilizar informação adicional acerca dos métodos e dos próprios problemas, que possa não ser usado de um modo puramente programático pelo sistema, mas que, contudo, possa ser útil para outros fins, nomeadamente para fins didácticos, historial e de

modo a permitir providenciar informação adicional sobre os conceitos tratados no sistema permitindo, deste modo, um melhor apoio à actividade de escalonamento.

6.3.2.1 Pesquisa de problemas

A pesquisa de problemas permite obter informação diversa sobre classes de problemas de escalonamento, incluindo uma descrição sucinta destas. Para que seja possível proceder à pesquisa de problemas é necessário que, previamente, tenha decorrido uma correcta especificação destes, tal como se descreveu e ilustrou nas secções 6.3.1.1 e 0. Sendo assim, quando um utilizador do sistema proceder à definição do problema que está interessado pesquisar desencadear-se-á um processo de pesquisa através da rede de nós pertencentes à OV. O nó que recebe o pedido se não possuir características de um nó especial, conhecendo a lista de nós activos na rede OV, terá de começar por pedir essa lista a um dos nós especiais, de modo a poder propagar o pedido aos restantes nós, membros da organização, em conformidade com o previamente descrito na secção 0. A Listagem 6.7 ilustra este pedido enviado em formato XML:

```
<activePeersListRequest>
        <peer>
                <name>… </name>
                <organization>… </organization>
                <address>http://…</address>
                <port>… </port>
                <restrict_domain>… </restrict_domain>
                <status>… </status>
        </peer>
</ activePeersListRequest>
```

Listagem 6.7 – Pedido de lista de nós activos na OV.

Em resposta a este pedido, um dos nós especiais envia a lista de nós activos, com os endereços correspondentes, como se ilustra na Listagem 6.8.

```
<activePeersListResponse> //peersList
        <peer>
                <name>… </name>
                <organization>… </organization>
                <address>http://…</address>
                <port>… </port>
                <restrict_domain>… </restrict_domain>
                <status>active</status>
        </peer>…
</ activePeersListResponse>
```

Listagem 6.8 – Lista de nós activos devolvidos pelo sistema.

De seguida é então possível enviar o pedido de classes relacionadas de problemas, que satisfazem as características especificadas pelo utilizador, através do envio de uma lista, em formato XML, de parâmetros relativos à caracterização do problema em questão a todos os

nós correntemente activos na OV, através de um processo de pesquisa na BCD. A Listagem 6.9 ilustra este tipo de lista, onde os parâmetros dos problemas estão expressos na forma de conteúdo das marcas "params", etc. em conformidade com a nomenclatura de classificação dos problemas de escalonamento (secção 4.4).

```
<allPeersRequest>
        <ActivePeers><address>http://…</address>…
        </ActivePeers>
        <params>
                <systemType>…</ systemType>
                <mpMachine>…</ mpMachine>
                <mpTask>…</mpTask>
                <machines>…</machines>
                <preemption>… </preemption>
                <precedences>… </precedences>
                <arrivals>… </arrivals>…
        </params>
</allPeersRequest>
```

Listagem 6.9 – Especificação de parâmetros dos problemas.

Em resposta a esta solicitação os nós activos criam uma lista de classes que são obtidas pela associação da lista de parâmetros recebidos, especificados pelo utilizador para o problema, com os parâmetros definidos para cada classe de problema constante no documento XML de problemas da BC de cada nó da OV (BCD). Uma vez obtidas correspondências entre a lista de parâmetros recebidos e os contidos no documento XML de problemas, de cada nó da rede OV, o nó que recebeu o pedido do utilizador procede à compilação de todos os resultados obtidos dos restantes nós numa especificação XML de resultados, como se ilustrada na Listagem 6.10.

```
<allPeersReplies>
        <SinglePeerReplies>
                <peerAddress>http://…</peerAddress>
                <problems>
                        <problem>
                                <problemClass>…</problemClass>
                                <complexity>…</complexity>
                                <context>…</context>
                                <params>…</params>
                        </problem>
                        …
                </problems>
        </SinglePeerReplies>
        …
</allPeersReplies>
```

Listagem 6.10 – Lista de problemas relacionados devolvidos pelo sistema.

Ilustração da pesquisa de problemas

Uma vez definido um problema pela utilizador do sistema e recolhidos os dados de todos os nós activos da OV relativos às classes de problemas constantes na BCD, através de um

processo de pesquisa de problemas como descrito na secção anterior, o resultado da pesquisa pode ser apresentado em diferentes formatos e um exemplo de uma descrição sucinta de classes de problemas é apresentado na Figura 6.11.

fm/FP,2|n|Cmax	Johnson (1954)	Maximal polynomially solvable. Without preemption.
fm/FP,2|n,rj|Cmax	Lenstra et al (1977)	Minimal NP-hard. Without preemption.
fm/FP,2|n,rj, no-wait|Cmax	Roeck (1984)	Maximal polynomially solvable. With preemption.
fm/FP,3|n,pmtn| Cmax	Gonzales and Sahni (1978). Cho and Sahni (1981).	Maximal polynomially solvable. With preemption
fm/FP,3|n|Cmax	Garey et al (1976).	Minimal NP-hard. Without preemption
fm/F,m|n,pji,prec| Cmax	Leung et al (1984). Timkovsky (1998).	Minimal NP-hard. Without preemption

Figura 6.11 – Classes de problemas de escalonamento da produção.

6.3.2.2 Pesquisa de métodos

Muitos métodos podem ser mais ou menos adequados para resolver uma determinada classe de problemas e na base de conhecimento de métodos (BCD) o sistema pesquisa os métodos que podem ser usados para resolver uma determinada classe de problemas. Esta pesquisa de métodos (para uma ou várias classes relacionadas de problemas) é realizada através de um mecanismo de associação de características dos problemas a características dos métodos.

Mecanismo de pesquisa - interacção entre problemas e métodos

A selecção de um método de escalonamento da produção apropriado a um determinado problema depende das características intrínsecas dos problemas, incluindo o critério de optimização para avaliar a qualidade das soluções obtidas, em conformidade com o que foi referido no capítulo 3.

Na literatura existem diversos tipos de métodos e neste trabalho o interesse recai, essencialmente, naqueles que sejam conhecidos como sendo aplicáveis para a resolução de problemas de escalonamento da produção.

No sentido de associar métodos a problemas torna-se necessário classificar os métodos e reconhecer a sua adequabilidade para a resolução de certas classes de problemas. Este processo pode ser conseguido, de um modo mais "classificado" na medida em que com base em conhecimento prévio já será possível saber se um determinado método resolve uma

determinada classe de problema ou não ou, simplesmente, tentar "encaixar" ou fazer corresponder problemas nas entradas de métodos, de algum modo sistemático e desejável.

A selecção de métodos para a resolução de problemas de escalonamento da produção requer portanto a identificação de características dos problemas de escalonamento, que permitam estabelecer uma associação aos métodos, isto é, características relacionadas com o tipo de sistema de produção em que o problema ocorre e um conjunto mais ou menos alargado de condições ou restrições relevantes impostas no problema, bem como a especificação da critério de optimização correspondente, tal como se descreveu nas secções anteriores (6.3.2.1).

As características "comuns" a problemas e a métodos reflectem ou traduzem os parâmetros de classificação dos problemas de escalonamento da produção que integram a nomenclatura de classificação proposta, apresentada no capítulo 4 e que têm correspondência com as especificações das entradas dos métodos implementados para resolver estes problemas.

A Figura 6.12 ilustra a metodologia que serve de suporte ao desenvolvimento do sistema demonstrador de apoio ao escalonamento da produção, que permite a pesquisa de método(s) para a resolução de problemas especificados.

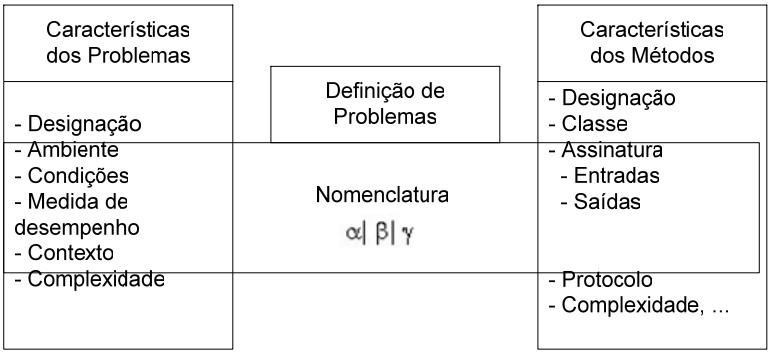

Figura 6.12 - Interacção entre características dos problemas e dos métodos.

Este esquema integra dois blocos de características fundamentais, que correspondem, respectivamente, às características dos problemas e às características dos métodos de escalonamento da produção. Estes blocos de características de problemas e de métodos além das características «comuns» a ambos (problemas e métodos) incluem ainda características próprias a problemas e a métodos, respectivamente, que por não serem relevantes para atingir o objectivo de associação de métodos a problemas não são consideradas no processo de associação de métodos a problemas, aquando da pesquisa de métodos de escalonamento da produção, para a resolução dos problemas colocados. São

exemplo, a descrição do contexto e a complexidade dos métodos e a identificação do protocolo de comunicação e da complexidade dos métodos.

O processo de pesquisa de métodos na BCD é efectuado através do módulo (MPC), anteriormente descrito na secção 6.3, através de um motor de pesquisa em *Prolog*, por cada nó da rede da organização virtual (OV). Este módulo contém um mecanismo de inferência interno, desenvolvido usando a ferramenta SWI-Prolog V.5.2.1, disponível em http://www.swi-prolog.org/.

Os métodos estão geralmente disponíveis na BC dos nós pertencentes à OV, mas podem também ser encontrados em outros sítios não pertencentes à comunidade.

Uma das funcionalidades do *Prolog* úteis que favoreceu a sua opção é o facto de sistemas em *Prolog* possuírem a capacidade de fazer inferências, mesmo com dados incompletos ou "imprecisos".

O *Prolog* é uma das linguagens lógicas mais conhecidas e utilizadas.

É uma linguagem que foi desenvolvida na Universidade de Marselha, como instrumento prático de programação em lógica. Trata-se de uma linguagem essencialmente útil para aplicações que envolvem processamento simbólico e/ou pesquisa automática de alternativas, ou uma pesquisa sistemática de soluções possíveis. Além disso, é uma linguagem relativamente simples e surpreendentemente potente, baseada em lógica de predicados de 1ª ordem, que permite escrever programas claros, legíveis, concisos e também cada vez mais eficientes e rápidos (Varela, 2000b).

O mecanismo de pesquisa aplicado em cada nó da rede da organização (OV) selecciona, de cada BC local da BCD, o(s) método(s) que podem ser utilizados para resolver um determinado problema. Para tal torna-se necessário proceder à troca de pedidos e respostas entre nós, começando pela identificação da(s) classe(s) de problema(s), descrita na secção 6.3.2.1, o que é efectuado em rede, (com base no protocolo HTTP). Os dados relativos às formulações dos pedidos e das respostas de pesquisa são transmitidos em formato XML entre os nós activos na rede P2P da OV.

A pesquisa de métodos é realizada através de um processo de duas fases: a primeira de associação de parâmetros de especificação de problemas, definidos pelo utilizador do sistema, e as especificações de problemas conhecidas (existentes nos repositórios XML de problemas, de cada nó da OV), tal como se descreveu anteriormente na secção 6.3.2.1 e a segunda da subsequente associação entre estas especificações de problemas e as

especificações de métodos (contidas nos repositórios XML de métodos, de cada nó da organização (OV) na rede P2P).

Os resultados da pesquisa de classes de problemas constituem então os resultados de uma primeira fase da pesquisa de métodos e consistem, portanto, na especificação de um conjunto de classes de problemas (<classe>… </classe>), obtidas de cada nó participante na pesquisa, que são compilados conjuntamente, de modo a serem devidamente formatados para serem apresentados ao utilizador do sistema que os solicitou. Ou seja, o mecanismo de pesquisa identifica uma lista de classes de problemas na BCD que satisfaça as características do problema, definidas pelo utilizador. Esta lista é apresentada ao utilizador, a fim de este poder identificar o tipo de problema a resolver. Este mecanismo de derivação de diferentes classes de problemas que satisfazem os parâmetros especificados pelo utilizador tem particular vantagem para identificar métodos que resolvem classes mais específicas e outros para resolver classes mais gerais, que englobam o problema específico e portanto o podem também resolver.

De seguida, após o utilizador ter seleccionado da lista o(s) problema(s), usando esta(s) class(es) seleccionada(s), um segundo pedido é submetido aos correspondentes nós, que constitui um pedido para encontrar métodos para este(s) problema(s), dando início a segunda fase do processo de pesquisa, através do envio da lista de classe(s) de problema(s) seleccionada(s), aos nós activos da rede OV. A Listagem 6.11 ilustra esta lista que, mais uma vez é enviada em formato XML.

```
<allPeersRequest>
        <ActivePeers>
                <address>http://…</address>…
        </ActivePeers>
        <slectedClasses>
                <problemClass>…</problemClass>
                <problemClass>…</problemClass>
                …
        </ slectedClasses>
</allPeersRequest>
```

Listagem 6.11 – Especificação do pedido de métodos para os problemas seleccionados.

Desencadeia-se então um processo similar ao que foi descrito para a primeira fase. Os (mesmos) nós activos da organização recebem a lista XML com as especificações da(s) classe(s) que o utilizador seleccionou e cada nó vai à procura, na sua BC local, de métodos disponíveis para essas classes (que incluem os parâmetros do problema previamente especificados pelo utilizador). Isto é, portanto, efectuado através de uma análise de correspondência entre classes de problemas da lista recebida e as classes de problemas

constantes no repositório XML de métodos de cada nó, de acordo com o conjunto de características especificadas pelos utilizadores para o(s) problema(s), sendo o resultado da pesquisa de cada um destes nós enviado ao nó que enviou o pedido, a fim de, mais uma vez, os resultados de todos os nós puderem ser reunidos e compilados conjuntamente para serem formatados e apresentados ao utilizador final.

A Listagem 6.12 ilustra a especificação XML correspondente aos resultados obtidos dos nós intervenientes na pesquisa que contém, portanto, a lista de métodos obtidos durante a segunda fase de pesquisa:

```
<allPeersReplies>
        <SinglePeerReplies>
                <peerAddress>http://…</peerAddress>
                <methods>
                        <method>
                                <Id>…</Id>
                                <methodClass>…</methodClass>
                                <methodName>…</methodName>
                                <url>…</url>
                                <reference>…</reference>
                                <complexity>…</complexity>
                                <protocol>…</protocol>
                        </method>…
                </methods>
        </SinglePeerReplies>…
</allPeersReplies>
```

Listagem 6.12 – Lista de métodos dos peers para o(s) problema(s) especificados.

Ilustração da pesquisa de métodos

Uma vez definidas as características do problema a resolver é então possível obter uma lista de classes de problemas relacionados. Os resultados podem ser visualizados através de uma tabela, tal como se ilustra na Figura 6.13. onde as referidas características especificadas para o(s) problema(s) podem ser observadas no topo da figura.

Problem types for the general problem characteristics: [system_type=F, jobs=n, measure=Cmax]

Class	Complexity	Context	Problem characteristics	Select
F2|n|Cmax	Maximal Polinomially Solvable	≥	[(system_type, F), (machines, 2), (jobs, n), (measure, Cmax)]	☑
F2|rj,n|Cmax	Minimal NP-hard	≥	[(system_type, F), (jobs, n), (arrivals, rj), (machines, 2), (measure, Cmax)]	☐
F2|rj,n,no-wai	Maximal Polinomially Solvable	≥	[(system_type, F), (machines, 2), (jobs, n), (arrivals, rj), (buffers, no-wait), (measure, Cmax)]	☐
F3|rj,n|Cmax	Minimal NP-hard	≥	[(system_type, F), (machines, 3), (jobs, n), (arrivals, rj), (measure, Cmax)]	☐
F3|n|Cmax	Minimal NP-hard	≥	[(system_type, F), (machines, 3), (jobs, n), (measure, Cmax)]	☑
Fm|n|Cmax	Minimal NP-hard	≥	[(system_type, F), (machines, m), (jobs, n), (measure, Cmax)]	☑
Fm|prec,pji=1,	Minimal NP-hard	≥	[(system_type, F), (machines, m), (jobs, n), (precedences, prec), (times, pji=1), (measure, Cmax)]	☐

Figura 6.13 - Selecção de problemas.

O utilizador pode seleccionar uma ou mais classes de problemas, especialmente, se não estiver completamente certo acerca do problema que pretende resolver.

Seguidamente, numa segunda fase do processo de pesquisa é então possível obter uma lista de métodos, obtidos através da BCD, que pode então ser apresentada em forma de tabela, tal como se apresenta na Figura 6.14.

Esta tabela apresenta informação diversa, que inclui referência a autores e referências bibliográficas, detalhes para ajudar na selecção de métodos adequados à resolução do problema, os endereços para a execução dos métodos disponíveis e outra informação genérica, nomeadamente, a classe a que cada método pertence e detalhes acerca de cada classe de problema, entre outra informação.

Available methods for solving the problem classes: [class=F2|n|Cmax, class=F3|n|Cmax, class=Fm|n|Cmax]

ID	Prob. Class	Method name	Implementation location	Reference	Complexity	Protocol	Select
1001	Fm|n|Cmax	BranchBoundIS	http://localhost:6002/RPC2	Ignall and Schrage, 1965	NP-hard	XML-RPC	◉
32	F2|n|Cmax	Johnson	http://localhost:6002/RPC2	Johnson, 1954	Polynomial	XML-RPC	○

Figura 6.14 - Métodos para a resolução de problemas.

Como ilustrado, o sistema possui um método disponível, que o utilizador seleccionou, para a resolução de problemas do tipo fm/FP,m|n|Cmax (ou, Fm|n|Cmax). Este método é o método *Branch-and-Bound* (B&B) de Ignall e Schrage (Ignall and Schrage 1965, Baker 1974).

Além disso, também se aparece disponível o método de Johnson (Johnson 1954, Baker 1974) para a resolução de problemas fm/FP,2| n| Cmax (ou, F2|n|Cmax), que constitui, portanto, uma subclasse da classe fm/FP,m| n| Cmax (ou, Fm|n|Cmax).

Por detrás desta informação o sistema também providencia informação mais detalhada acerca dos problemas, dos métodos e das suas implementações, de modo a permitir uma mais fácil selecção de métodos. Por exemplo, informação adicional sobre a complexidade do método que, neste caso do método B&B de Ignall-Schrage, é um método de programação matemática exacta, com complexidade temporal exponencial é informação útil que também é possível ser consultada.

Se não houver informação disponível na BCD, sobre métodos para a resolução de um determinado problema especificado, torna-se necessário repetir o processo de pesquisa, através da re-especificação de características do problema, de modo a tentar encontrar método(s) adequado(s). Sendo assim, o utilizador pode tentar "relaxar" algumas das

restrições do problema, simplificando-o, ou introduzir novas características e repetir o processo de pesquisa, as vezes que quiser.

6.3.3 Invocação de métodos

O processo de invocação de métodos é realizado pelo módulo MIM, tal como se explicou na secção 6.3, e foi projectado e implementado na forma de um serviço de web.

Este serviço pode ser acedido através de http://www.dps.uminho.pt/ e o seu funcionamento é descrito a seguir, recorrendo ao protocolo XML-RPC (*extensible markup language – remote procedure call*) (Varela, 2003).

Serviço web

A expressão "serviço de web" ou simplesmente "serviço web" (do inglês, *web service*) emergiu como uma categoria genérica para designar serviços baseados na web fracamente acoplados e dinamicamente conectados, que integram um conjunto de ferramentas que permitem construir aplicações com base em infra-estruturas de web existentes (http://www.w3.org).

Num serviço web um determinado método aceita como entrada a definição de um problema e devolve um resultado numa determinada forma. A principal motivação para a utilização de serviços de web neste trabalho consiste em ser capaz de providenciar um serviço que:

> *para um determinado conjunto de parâmetros, que descrevem um determinado problema de escalonamento da produção colocado (uma instância de problema), retorne um conjunto de resultados, passíveis de serem aceites como uma solução ou uma resposta para o problema.*

Através deste serviço um cliente, de cada vez que recebe os dados de uma instância de problema a ser resolvida, por um dado método, previamente seleccionado pelo utilizador do sistema, envia um pedido para a invocação desse método, com os referidos dados do problema, ao processador (sistema) que o disponibiliza e este retorna a resposta ao cliente (nó que solicitou o serviço), com os resultados do problema, de modo a poderem ser apresentados ao utilizador, num determinado formato desejado. A Figura 6.15 ilustra o funcionamento do serviço web.

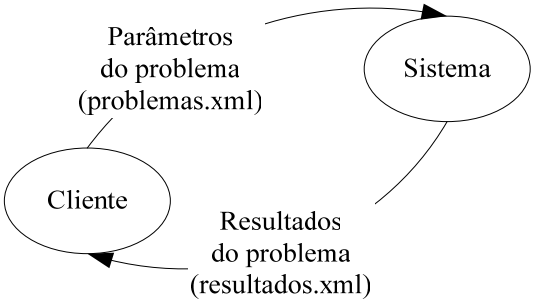

Figura 6.15 – Serviço de web.

Diferentes implementações de métodos podem providenciar diferentes tipos de resultados, por isso o sistema terá de possuir a sua descrição, de modo a poder formatá-los, de acordo com a forma que seja usada para apresentar os resultados do problema ao cliente, como último passo do serviço.

Além disso, as interfaces para introdução de dados do problema são automaticamente geradas para a execução de cada método particular e de um modo ajustado à dimensão e características específicas de cada par problema/ método. Sendo assim, o sistema terá de conhecer exactamente as assinaturas dos métodos remotamente implementados, em termos de descrição das suas entradas e saídas e dispor de folhas de estilo XSL apropriadas à formatação das páginas a serem apresentadas através de um browser de web para a introdução dos dados e apresentação dos resultados dos diferentes métodos, tal como se descreveu e ilustrou previamente na secção 6.3.1.3.

Toda a informação, relativa a problemas e a métodos está descrita nos respectivos repositórios XML e correspondentes DTDs (Varela 2002a and 2002b), que é introduzida no sistema através do módulo de inserção de novo conhecimento (MIC), aquando da especificação de novos problemas e métodos no sistema, para integrar a BCD, para futuras solicitações de pesquisa de informação e de execução dos métodos, como foi previamente explicado e ilustrado na secção 6.3.1.

No entanto, o resultado da execução de um método implementado para a resolução de uma determinada instância de problema é sempre transmitido ao cliente na forma de um ficheiro XML e o processamento e formatação subsequente destes dados é posteriormente efectuada localmente, em cada nó da OV, através dos componentes comuns do sistema.

Estes serviços de web podem ser usados para invocar serviços ou métodos em aplicações remotas usando interacções do tipo RPC (*Remote Procedure Call*) implementadas usando apenas mensagens XML (Carlson, 2001).

Protocolo XML-RPC

A invocação remota de métodos é ilustrada, no demonstrador SWEP, através de um serviço de web, usando o protocolo XML-RPC.

O protocolo XML-RPC (*XML-remote procedure call*) é o protocolo de comunicação usado para a invocação remota de métodos (Laurent et al. 2001, Varela et al 2002).

Este protocolo consiste numa sequência e estrutura de pedidos ou solicitações e respostas necessárias para levar a cabo o processo de invocação de comunicações num processador remoto, usando apenas mensagens em XML (Laurent, 2001; Varela, 2003b).

O mecanismo de chamadas remotas (*RPC – remote procedure call*) é uma forma usual para implementar comunicações entre um cliente e o servidor de aplicações distribuídas (Pires, J. N., 2002). O cliente faz aquilo que se assemelha a uma chamada local, embora o recurso/método não seja local. O mecanismo de XML-RPC transforma essa chamada numa chamada em rede adicionando o necessário para estabelecer essa comunicação. O servidor (nó que está a funcionar como servidor) recebe o pedido e executa-o, de acordo com a parametrização fornecida e devolve as respostas.

A invocação de métodos é então suportada pela linguagem XML, que é utilizada como meio de comunicação para invocar os métodos acessíveis através da Internet. Sendo assim, os dados relativos aos problemas são passados aos métodos em formato XML e o mesmo acontece aos resultados/ respostas dos métodos que, uma vez obtidos, são transferidos, via web, ao cliente, de modo a puderem ser formatados e apresentados ao utilizador que os solicitou.

O serviço de web, através do protocolo de comunicação XML-RPC, usa XML para codificar tanto o invólucro da mensagem como o conteúdo do corpo da mensagem. Como resultado, obtém-se um ambiente heterogéneo, em que cada componente participante no serviço (cliente que solicita o serviço e servidor de métodos) pode usar a sua própria tecnologia, sendo a integração completamente independente, nomeadamente, da plataforma/ sistema operativo e da linguagem de programação em que os métodos estão implementados e ainda de outras tecnologias intermédias (http://www.xmlrpc.com).

O único requisito fundamental é o de que cada componente tenha a capacidade de processar documentos XML e que cada nó ligado num ambiente distribuído suporte HTPP como camada de transporte por defeito.

Outros protocolos e tecnologias podiam igualmente ser utilizados, nomeadamente, SOAP (*Simple Object Access Protocol*), UDDI (*Universal Description, Discovery, and Integration of business for the web*), WSDL (*Web Services Description Language*), ou outros também bem conhecidos, como CORBA (*Common Object Request Broker Architecture*), RMI (*Remote Method Invocation*) ou DCOM (*Distributed Component Object Model*).

Ilustração da invocação de métodos

Na sequência do exemplo que tem vindo a ser considerado, nas secções anteriores, para ilustração do funcionamento deste sistema considere-se a instância de problema apresentada na secção 6.3.1.1, pertencente à classe fm/FP,m|n|Cmax e a lista de métodos disponíveis para a resolver, como apresentado na Figura 6.11, da secção 6.3.2.1. Nessa figura havia apenas um método susceptível de ser usado para resolver a instância de problema em questão, que era o método *Branch and Bound* de Ignall e Schrage (Ignall and Schrage, 1965; Baker, 1974, que pertence à classe dos métodos de programação matemática exacta, usando a técnica B&B, com complexidade temporal exponencial implementado em C++ e a correr sob o protocolo XML-RPC.

Através do módulo (MIM), sempre que um utilizador do sistema selecciona um determinado método, local ou remotamente disponível em rede, para resolver um determinado problema, desencadeia um processo de geração automática de interfaces para a introdução de dados desse problema, bem como para a apresentação dos resultados obtidos da execução do método, o que é realizado pelo módulo de interface com o utilizador (MIL), que se baseia nas especificações das assinaturas dos métodos contidas no sistema, nos respectivos documentos XML e nas correspondentes especificações DTD, bem como em folhas de estilo XSL, como previamente explicado e ilustrado.

Uma vez seleccionado o método B&B de Ignall-Schrage, neste caso, desencadear-se-á o processo para a introdução dos dados concretos da instância de problema a resolver.

A Figura 6.16 ilustra a interface para inserção de dados do problema em questão, que numa primeira fase, mais concretamente, consiste na introdução do número de trabalhos a processar e do número de processadores que integram a linha de produção.

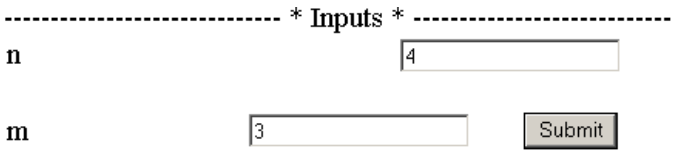

Figura 6.16 - Dados gerais do problema.

De seguida, o sistema gera, uma nova interface, com as células apropriadas para a introdução dos restantes dados do problema, necessários à execução do método, que neste caso são apenas os tempos de processamento dos trabalhos nos diferentes processadores da linha de produção.

i\j	1			2		
*	job	mach	p	job	mach	p
1	L1	M1	3	L2	M1	11
2	L1	M2	4	L2	M2	1
3	L1	M3	10	L2	M3	5

…

Figura 6.17 - Dados específicos do problema.

Seguidamente, uma vez obtidos os dados e a confirmação do utilizador para a execução do método, o sistema envia um pedido de invocação de método, através do protocolo XML-RPC, ao processador que disponibiliza o método a executar. Este processador pode ser um dos nós activos da rede OV ou pode ser uma qualquer outro processador não pertencente à organização. O método pode ainda estar a correr localmente no próprio processador que recebeu o pedido.

A linguagem XML providencia um vocabulário para descrever invocações, que consistem em chamadas remotas a procedimentos ou métodos que são transmitidas entre computadores usando o protocolo HTPP.

Clientes XML-RPC efectuam pedidos a procedimentos aos servidores XML-RPC que, por seu lado, devolvem resultados aos clientes XML-RPC.

Os clientes XML-RPC usam as mesmas funcionalidades HTTP que os clientes de browsers de web e os servidores XML-RPC usam as mesmas funcionalidades HTTP que servidores de web.

O protocolo XML-RPC requer um mínimo de cabeçalhos HTTP a serem enviados em conjunto com o pedido XML ao método para a resolução de uma determinada instância de problema (Varela, 2003). A Listagem 6.13 ilustra um pedido, em formato XML-RPC, através de um exemplo de junção de cabeçalhos com conteúdo XML, formando um pedido XML-RPC completo ao método em consideração para a resolução da instância de problema introduzida.

```
POST /rpchandler HTTP/1.0
User-Agent: AcmeXMLRPC/1.0
Host:localhost:5001
Content-Type: text/xml
Content-Length: 832
<?xml version="1.0"?>
<methodCall>
<methodName>getExactBranchBound</methodName>
<params><param><value><int>4</int></value></param>
  <param><value><int>3</int></value></param>
  <param><value><array><data>
       <value><string>J1</string></value><value><string>M1</string></value>
       <value><double>3</double></value>…
</data></array></value></param>
</params></methodCall>
```

Listagem 6.13 - Um pedido XML-RPC.

Um servidor XML-RPC, uma vez tendo recebido um pedido, terá de enviar uma resposta ao cliente. A resposta pode tomar uma de duas formas: o resultado do processamento do método invocado ou um relatório de erro, a indicar que algo correu mal no tratamento do pedido vindo do cliente.

Tal como acontece com um pedido XML-RPC, a resposta consiste em cabeçalhos HTTP e um conteúdo XML. O resultado da execução de uma determinada implementação de um método para uma dada instância de problema é então enviado ao cliente em formato XML.

A Listagem 6.14 apresenta uma resposta completa de um servidor XML-RPC, para a instância de problema considerada, incluindo ambos, cabeçalhos HTTP e o conteúdo XML.

```
HTTP/1.0 927 OK
Date: Wed, 22 Oct 2003 09:32:07 GMT
Server: MyCustomXMLRPCserver
Connection: close
Content-Type: text/xml
Content-Length: 871
<?xml version="1.0" encoding=" UTF-8"?>
<methodResponse>
<params>
  <param><value><string>J1,J3,J4,J2</string></value></param>
  <param><value><double>39</double></value></param>
  <param><value>
    <array><data>
       <value><string>J1</string></value>
       <value><string>M1</string></value>
       <value><double>0</double></value>
       <value><double>3</double></value>…
     </data></array>
</value></param></params></methodResponse>
```

Listagem 6.14 – Uma resposta XML-RPC.

O sistema permite diferentes formas de apresentação e armazenamento de resultados.

Se tudo tiver corrido bem com a execução do método, o sistema automaticamente gera uma interface gráfica para apresentar os resultados, novamente de acordo com a assinatura do método pré-definida no sistema. A Figura 6.18 ilustra a interface relativa à apresentação dos resultados do problema através de uma tabela, no browser do cliente, contendo os tempos de início e de fim de cada trabalho, em cada processador da linha, para o problema em questão.

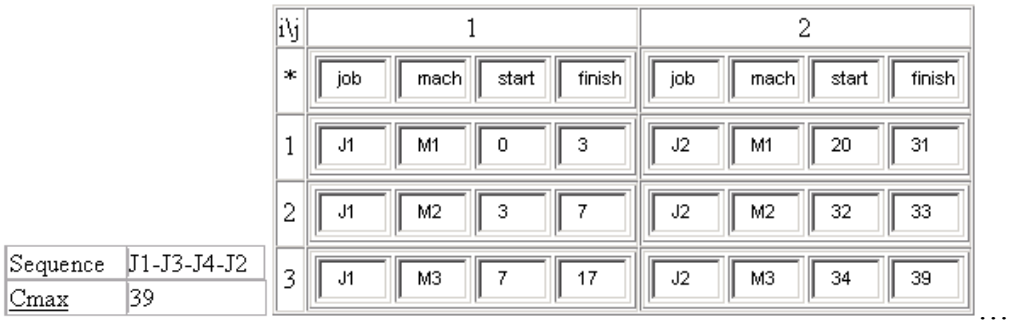

Figura 6.18 - Apresentação de resultados do problema

Podem ser usadas outras formas alternativas para apresentação de resultados dos métodos, incluindo apresentação directa de resultados, não transformados, em formato XML e podem ser transformados numa outra forma mais expressiva de apresentação de resultados, nomeadamente através de um diagrama de Gantt, que são também gerados automaticamente pelo sistema, a partir dos resultados obtidos pela execução do método, tal como se descreveu na secção 6.3.1.3. Isto é, mais uma vez, facilmente conseguido, dado os resultados estarem expressos em formato XML e existirem folhas de estilo XSL apropriadas, que permitem uma forma fácil de conversão de resultados dos métodos em diferentes apresentações desejadas.

6.4 Teste de avaliação do demonstrador

Nesta secção serão apresentados alguns problemas de escalonamento da produção industrial a fim de testar o sistema demonstrador proposto e, principalmente, ilustrar a utilidade que um sistema desta natureza na resolução destes problemas. Sendo assim, inclui-se uma análise de resultados, com o objectivo de avaliar a validade e utilidade do sistema.

De modo a cobrir diferentes cenários foram seleccionados problemas típicos, agrupados em três casos distintos, que são apresentados a seguir, bem como os resultados obtidos pela aplicação de alguns métodos disponíveis através do sistema para os resolver.

Sendo assim, o primeiro caso aborda quatro problemas, que ocorrem num mesmo tipo de ambiente de produção. Todos estes problemas possuem as mesmas características, incluindo exactamente os mesmos parâmetros de classificação dos trabalhos a processar (características da classe beta), diferindo apenas no critério de optimização (classe gama). Estes critérios são, contudo, de alguma forma relacionados entre si.

No segundo caso são abordados dois tipos de problemas, que diferem apenas na classe alfa, mais precisamente, no tipo de sistema de produção, mantendo-se, portanto, as características da classe beta e a critério de optimização.

Finalmente, o terceiro caso é relativo a problemas que diferem apenas em termos de características da classe beta, mantendo-se invariável o ambiente de produção e a critério de optimização a optimizar.

6.4.1 Caso 1 – Problemas de sistema de fase única e processador único

Problemas bastante conhecidos e estudados são problemas que ocorrem em ambiente de processador único. Estes problemas têm particular interesse dado que muitos problemas que ocorrem em ambientes complexos, tais como as oficinas de fabrico, podem e muitas vezes são decompostos e analisados através de subproblemas mais simples, nomeadamente problemas do tipo processador único.

A seguir são analisados quatro tipos destes problemas, que apenas diferem nos critérios de optimização que têm subjacentes. Estes critérios têm bastante interesse industrial real e estão relacionados com tempos de processamento e datas de entrega ou conclusão dos trabalhos e consideram ainda a possibilidade de atribuição de diferentes pesos ou urgências para a realização dos diversos trabalhos e a sua influência na tomada de decisão, face aos diferentes critérios de optimização consideradas em cada tipo de problema.

O primeiro problema considera a minimização do tempo de percurso médio dos trabalhos no sistema (Fmed) sendo, portanto, do tipo f1|n,dj,wj|Fmed. O segundo problema tem por objectivo a minimização do atraso médio dos trabalhos, Lmed (f1|n,dj,wj|Lmed). O terceiro problema consiste na minimização do atraso máximo dos trabalhos, Lmax (f1|n,dj,wj|Lmax) e, por fim, o quarto problema tem por objectivo a minimização da soma dos atrasos absolutos pesados, ΣwjTj (f1|n,dj,wj|ΣwjTj) dos trabalhos a processar no processador. Para todos os n trabalhos (Tj) são conhecidos os tempos de processamento (tj), as datas ou prazos de entrega ou conclusão (dj) e ainda os pesos ou prioridades ou urgências relativas de cada um deles (wj).

Este último (quarto) problema consiste numa generalização de um problema bem-conhecido que é o f1|n,dj|ΣTj. Diversas abordagens têm vindo a ser desenvolvidas para tentar resolver este problema. Tais abordagens variam desde técnicas sofisticadas, que requerem elevado tempo computacional até heurísticas simples. Também têm sido usadas abordagens baseadas em programação dinâmica e em técnicas de ramificação e limite (truncagem) (Pinedo, 2002).

Considerem-se os dados apresentados na Tabela 6.2 para uma instância de problema que irá ser considerada para os diferentes tipos de classes anteriormente descritas. Nesta instância existem quatro trabalhos (n=4), com os tempos de processamento (tj), as datas de entrega (dj) e as prioridades ou urgências dos trabalhos (wj) fornecidos. Pretende-se então minimizar, em cada caso, respectivamente, o tempo de percurso médio (Fmed), o atraso médio (Lmed), o atraso máximo (Lmax) e o atraso absoluto total pesado destes trabalhos no sistema (ΣwjTj).

Tabela 6.2 - Dados de instância dos problemas f1|n,dj|Fmed;Lmed;Lmax;ΣwjTj.

Trabalho	T1	T2	T3	T4
pj	10	10	13	4
dj	4	2	1	12
wj	14	12	1	12

Para resolver estes problemas que, portanto, apenas diferem no tipo de critério de optimização a minimizar (baseado em tempo), o sistema permite efectuar uma pesquisa através da sua base de conhecimento distribuída, a fim de descobrir métodos disponíveis adequados à resolução de cada tipo de problema particular em causa, bem como para resolver estas variantes do problema de base.

Sendo assim, para um mesmo problema de base, como o caso aqui apresentado, se o utilizador do sistema pretender testar a minimização de critérios de optimização diferentes mas proximamente relacionados, como neste caso, em que os critérios de optimização são todas relativas à minimização de critérios baseados no tempo: tempo de percurso médio (Fmed), atraso médio (Lmed), atraso máximo (Lmax) e atraso total absoluto pesado (ΣwjTj) o sistema disponibilizará diferentes tipos de métodos, para cada cenário particular, de acordo com as implementações de métodos que tem local ou remotamente acessíveis e disponíveis e que podem ser usadas para a resolução de cada tipo de problema em questão.

Os métodos actualmente disponíveis através do sistema que podem ser usados para a resolução destes problemas são:

- A regra de sequenciação SPT, para os problemas do tipo f1|n,dj,[wj]|Fmed e f1|n,dj,[wj]|Lmed;

- A regra de sequenciação EDD, para o problema do tipo f1|n,dj,[wj]|Lmax;

- O método EA (baseado num algoritmo evolutivo), para o tipo de problema f1|n,dj,wj|ΣwjTj.

Aplicando estes métodos é possível obter os resultados apresentados a seguir, que estão expressos através de diagramas de Gantt e de tabelas que resumem os principais resultados obtidos pela aplicação dos diferentes métodos aos problemas considerados.

Problemas f1|n,dj|Fmed e f1|n,dj|Lmed

Os problemas do tipo f1|n,dj,wj|Fmed e f1|n,dj,wj|Lmed podem ser ambos resolvidos, de uma forma muito simples e com garantia de obtenção de solução óptima, através da aplicação da regra de prioridade SPT (*Shortest Processing Time*), que consiste em atribuir os trabalhos ao processador por ordem crescente dos seus tempos de processamento (tj). Os resultados detalhados para cada trabalho, em termos de tempo de percurso de cada trabalho j (Fj), atraso de cada trabalho j (Lj) e atraso absoluto de cada trabalho (Tj), obtidos através da aplicação da regra SPT, para estes dois problemas, que se podem exprimir na forma de um único problema, na forma f1|n,dj,wj|Fmed,Lmed, estão apresentados na Tabela 6.3.

Tabela 6.3 – Resultados obtidos pela regra STP.

Trabalhos ordenados	T4	T1	T2	T3
Fj	4	14	24	37
Lj	-8	10	22	36
Tj	0	10	22	36

Deste modo, a sequência óptima (em termos de tempo de percurso médio e de atraso médio dos trabalhos) de processamento é T4, T1, T2, T3 e os valores obtidos para os critérios de optimização são: tempo de percurso médio, Fmed=19.75 T.U. (valor mínimo), atraso médio, Lmed=15 U.T. (valor mínimo), (atraso absoluto médio, Tmed=17 U.T.), atraso máximo, Lmax=36 U.T. e atraso absoluto total pesado, ΣwjTj=440. A Figura 6.19 apresenta o diagrama de Gantt para esta solução.

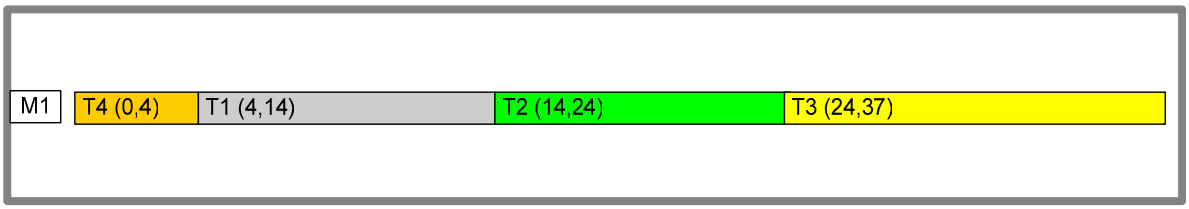

Figura 6.19 – Diagrama de Gantt: regra SPT.

Problema f1|n,dj|Lmax

O problema f1|n,dj|Lmax pode ser resolvido, em termos óptimos, através da aplicação da regra de prioridade ou despacho EDD (*Earlist Due Date*). Esta regra é uma regra de sequenciação simples, que consiste na ordenação dos trabalhos para serem processados no processador por ordem crescente das suas datas ou prazos de conclusão ou de entrega. Os resultados detalhados obtidos pela aplicação desta regra, para cada trabalho, em termos de Fj, Lj e Tj, para o problema f1|n,dj,wj|Lmax, estão apresentados na Tabela 6.4.

Tabela 6.4 – Resultados obtidos pela regra EDD.

Trabalhos ordenados	T3	T2	T1	T4
Fj	13	23	33	37
Lj	12	21	29	25
Tj	12	21	29	25

Deste modo, a sequência óptima de processamento dos trabalhos é T3, T2, T1, T4 e os valores obtidos para os critérios de optimização são: tempo de percurso médio, Fmed=26.5 T.U., atraso médio, Lmed=21.75 U.T. (atraso absoluto médio, Tmed=Lmed=21.75 U.T.), atraso máximo, Lmax=29 U.T. e atraso absoluto total pesado, ΣwjTj=970.

A Figura 6.20 representa o correspondente diagrama de Gantt para esta solução.

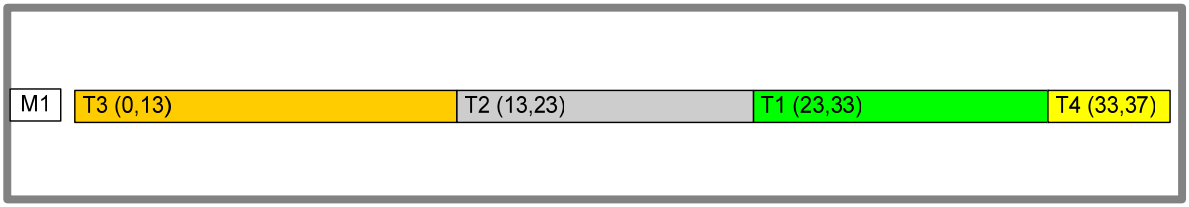

Figura 6.20 – Diagrama de Gantt: regra EDD.

Problema f1|n,dj|ΣwjTj

O problema f1|n,dj,wj|ΣwjTj pode ser resolvido, sem garantia de obtenção da solução óptima, entre muitos tipos de métodos, através de um método de pesquisa local, nomeadamente através de um método ou algoritmo evolucionário (Varela, 2004). Aplicando este método é possível obter os resultados detalhados para cada trabalho deste problema, em termos de Fj, Lj e Tj, apresentados na Tabela 6.5.

Tabela 6.5 – Resultados obtidos pelo método EA.

Trabalhos ordenados	T1	T4	T2	T3
Fj	10	14	24	37
Lj	6	2	22	36
Tj	6	2	22	36

Deste modo, a sequência de processamento dos trabalhos é T1, T4, T2, T3 e os valores obtidos para os critérios de optimização são: tempo de percurso médio, Fmed=21.25 T.U., atraso médio, Lmed=16.5 U.T. (atraso absoluto médio, Tmed=Lmed=16.5 U.T.), atraso máximo, Lmax=36 U.T. e atraso absoluto total pesado, ΣwjTj=408.

A Figura 6.21 representa o correspondente diagrama de Gantt para esta solução.

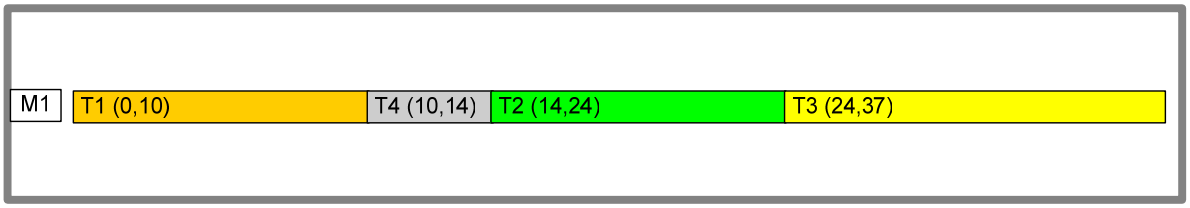

Figura 6.21 – Diagrama de Gantt: método EA.

A Tabela 6.6 resume os resultados obtidos pela execução dos três métodos, o método evolucionário (EA) para a minimização do atraso absoluto total pesado e as duas regras de sequenciação (SPT e EDD) para a minimização do atraso médio (e tempo de percurso médio) e atraso máximo, respectivamente. A negrito apresentam-se as melhores soluções obtidas pela aplicação de cada método.

Tabela 6.6 – Resultados obtidos pelos três métodos para f1|n,dj,wj|Fmed, Lmed;Lmax;ΣwjTj.

Sequência	Critério de optimização	Método

T4, T1, T2, T3	Fmed = 19,75 Lmed= 15 Lmax = 36 ∑wjTj = 440	SPT
T3, T2, T1, T4	Fmed = 26,5 Lmed = 21,75 Lmax = 29 ∑wjTj = 970	EDD
T1, T4, T2, T3	Fmed = 21,25 Lmed = 16,5 Lmax = 36 ∑wjTj = 408	EA

Analisando os resultados anteriormente apresentados na Tabela 6.6 pode constatar-se que o método EA permitiu obter a melhor solução em termos de atraso absoluto total pesado, enquanto que as regras de sequenciação, tal como esperado, mostraram um melhor desempenho em termos de atraso máximo (regra EDD) e de atraso médio e tempo de percurso médio (regra SPT), exprimindo soluções óptimas para os correspondentes critérios de optimização que minimizam.

Através destes exemplos simples é possível constatar que ligeiras diferenças nas formulações dos problemas, nomeadamente uma variação apenas no tipo de critério de optimização a considerar como objectivo do problema a resolver, podem resultar em classes de problemas distintas e, deste modo, requerer a aplicação de diferentes métodos, que deverão, portanto, estar disponíveis e facilmente acessíveis para a resolução de cada problema particular colocado pela comunidade do escalonamento da produção.

6.4.2 Caso 2 – Problemas de sistema de fases múltiplas de linha pura vs. oficina pura

Problema fm/FP,2|n|Cmax

Um outro problema muito interessante e bastante analisado na prática é o problema do tipo fm/FP,2|n|Cmax. Considere-se uma instância de problema com quatro trabalhos, que devem ser processados numa linha pura com dois processadores. O objectivo consiste em minimizar o tempo total ou instante máximo de percurso ou em curso dos trabalhos no sistema (Cmax). A Tabela 6.7 contém os tempos requeridos ao processamento de cada trabalho j em cada uma dos dois processadores i do sistema (tji).

Tabela 6.7 – Dados do problema fm/FP,2|n|Cmax.

i / j	T1	T2	T3	T4	T5	T6
m1	2	3	2	6	2	5
m2	4	1	5	3	4	7

Aplicando o método ou regra de Johnson (Conway, 1967), que é um método muito simples, baseado na regra de sequenciação SPT (capítulo 2, secção 2.4) é possível obter, de uma forma muito simples, uma solução óptima para o problema (neste caso para a instância de problema fm/FP,2|5|Cmax). A Tabela 6.8 mostra os resultados obtidos, em termos de instantes de início e de fim de cada trabalho em cada um dos processadores. O valor óptimo da critério de optimização é Cmax=26 unidades de tempo e a sequência óptima de fabrico dos trabalhos é T5, T3, T1, T6, T4, T2.

Tabela 6.8 – Resultados do problema.

i / j		T1	T2	T3	T4	T5	T6
m1	início	4	17	2	11	0	6
	fim	6	20	4	17	2	11
m2	início	11	25	6	22	2	15
	fim	15	26	11	25	6	22

A Figura 6.22 ilustra o correspondente diagrama de Gantt para o problema.

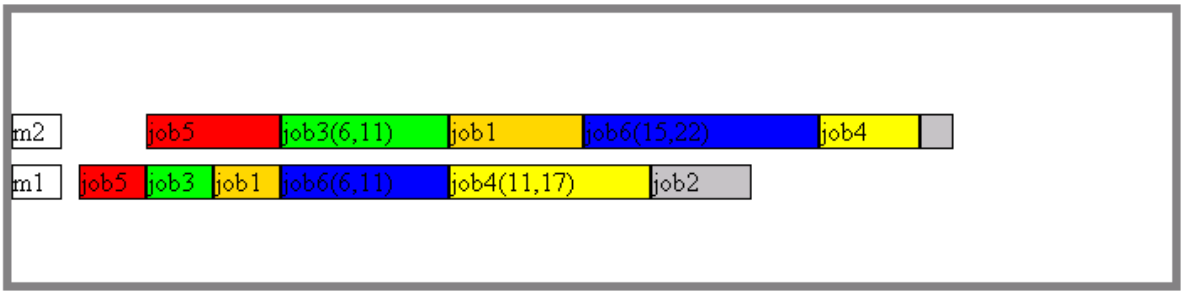

Figura 6.22 – Diagrama de Gantt: regra de Johnson.

Problema fm/JP, 2|n|Cmax

Considere-se agora que os mesmos trabalhos referidos na anterior secção 0 (com os mesmos dados) passam a ser processados numa oficina pura, também com dois processadores, em vez de numa linha simples e que o objectivo continua a ser o da minimização do tempo total de fabrico destes lotes no sistema (Cmax). Para uma instância de problema, portanto, com quatro trabalhos e os mesmos tempos de processamento. A Tabela 6.9 contém os tempos requeridos ao processamento de cada trabalho j em cada um dos dois processadores i por onde os trabalhos podem passar, neste caso A ou B (tji).

Tabela 6.9 – Dados do problema.

j	T1	T2	T3	T4	T5	T6
1ª op.	3 | A	5 | A	6 | B	6 | B	2 | A	5 | B
2ª op.	4 | B	---	---	3 | A	4 | B	7 | A

Aplicando agora o método de Jackson (Silva, 2003), obtêm-se os resultados apresentados na Tabela 6.10, que apresenta os instantes de início e de fim de cada um dos trabalhos em cada um dos processadores requeridos ao seu processamento (operação 1 e, em alguns casos também a operação 2, a serem processadas em uma das duas maquinas A ou B da oficina). O tempo total ou máximo dos trabalhos na oficina é agora de 25 U.T. e as sequências óptimas de fabrico destes trabalhos em cada um dos dois processadores da oficina, A e B, são as seguintes:

Processador A: T5, T1, T2, T6, T4.

Processador B: T6, T4, T3, T5, T1.

Tabela 6.10 – Resultados do problema.

j		T1	T2	T3	T4	T5	T6
1ª op.	início	A | 2	A | 5	B | 11	B | 5	A | 0	B | 0
	fim	A | 5	A | 10	B | 17	B | 11	A | 2	B | 5
2ª op.	início	B | 21	---	---	A | 17	B | 17	A | 10
	fim	B | 25	---	---	A | 20	B | 21	A | 17

A Figura 6.23 ilustra o correspondente diagrama de Gantt para o problema.

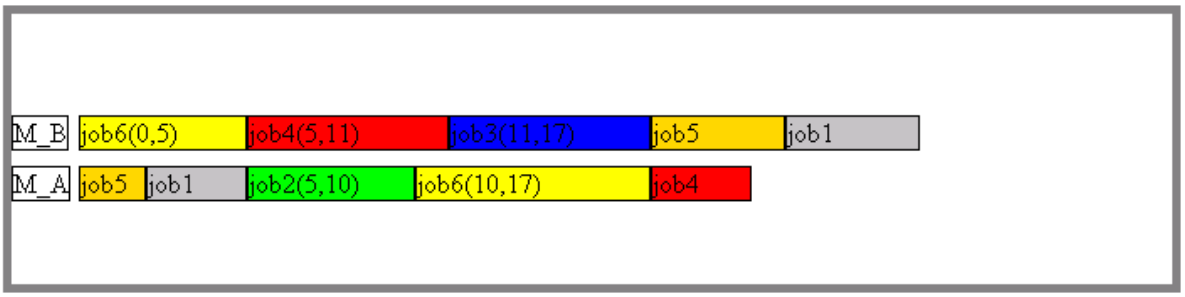

Figura 6.23 – Diagrama de Gantt: método de Jackson.

6.4.3 Caso 3 – Problemas de sistema de fase única de processadores paralelos

Problema f1/PI,m|n,pmtn|Cmax

Os problemas que ocorrem em ambiente de processadores paralelos constituem mais um tipo de problemas que tem merecido a atenção de muitos investigadores ao longo das últimas décadas. Um exemplo bem conhecido e tratado deste tipo de problemas é o problema f1/PI,m|n,pmtn|Cmax, relativo ao processamento de um conjunto de n trabalhos, que permitem interrupção no seu processamento, o que é expresso pela presença da característica pmtn, e que consistem, portanto, numa única operação, que poderá ser realizada em uma de um conjunto de m=4 processadores paralelos disponíveis. O objectivo consiste, mais uma vez, em minimizar o instante máximo de percurso dos trabalhos no sistema (Cmax), uma dos critérios de optimização mais objectivadas do escalonamento em geral. A Tabela 6.11 apresenta os tempos de processamento para realizar a operação de cada um de quinze trabalhos.

Tabela 6.11 – Dados do problema.

Trabalhos	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
tj	5	10	14	7	4	6	2	3	5	7	8	1	6	9	5

Aplicando o método de McNaughton (Conway, 1967) é possível obter uma solução óptima para o problema e a Tabela 6.12 apresenta a solução obtida pela aplicação do referido método, indicando o instante de início e de fim de cada um dos trabalhos e o processador em que cada um deve ser realizado.

Tabela 6.12 – Resultados do problema.

Trab. j	T1	T2	T3	T3	T4	T5	T6	T7	T8	T9	T10	T11	T11	T12	T13	T14	T15
Máq. i	M1	M1	M1	M2	M2	M2	M2	M3	M3	M3	M3	M3	M4	M4	M4	M4	M4
Início	0	5	15	0	6	13	17	0	2	5	10	17	0	2	3	9	18
Fim	5	15	23	6	13	17	23	2	5	10	17	23	2	3	9	18	23

O tempo total dos trabalhos no sistema de processadores paralelos é de 23 U.T. e sequências de fabrico possíveis dos quinze trabalhos em cada uma das quatro processadores, M1 a M4, são as seguintes:

Processador M1: T1, T2, T3.

Processador M2: T3 (parte restante), T4, T5 e T6.

Processador M3: T7, T8, T9, T10 e T11.

Processador M4: T11 (parte restante), T12, T13, T14 e T15.

A Figura 6.24 ilustra o correspondente diagrama de Gantt para o problema.

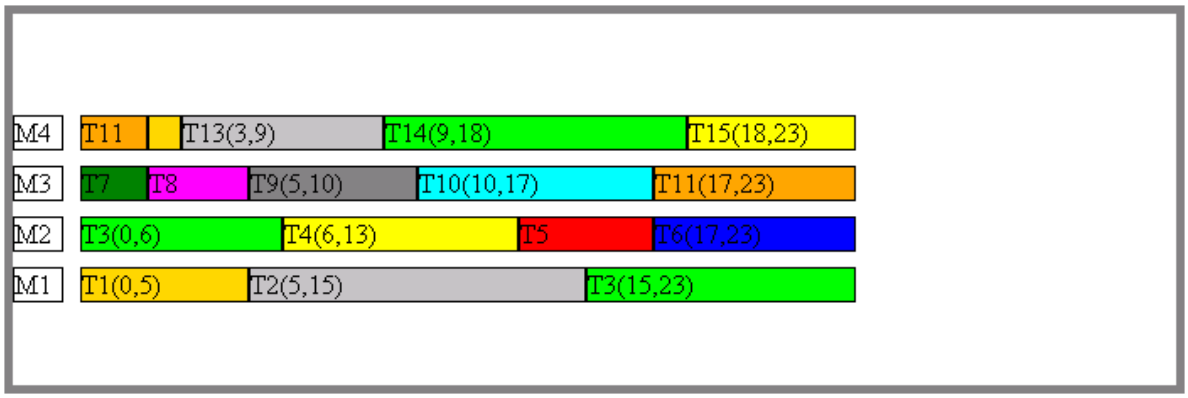

Figura 6.24 – Diagrama de Gantt: método de McNaughton.

Problema f1/PI,m|n|Cmax

Considerando agora uma variação nas características do problema anterior, ou seja supondo agora o cenário em que os problemas a processar nos processadores paralelos não permitem interrupção no seu processamento o problema torna-se mais complexo e passa a ser representado pela classe f1/PI,m|n|Cmax.

Se continuarmos a ter os quinze trabalhos com os mesmos tempos de processamento da operação requerida, em uma das quatro processadores paralelos disponíveis e mantendo-se também o objectivo da minimização do tempo total de percurso destes no sistema, torna-se necessário seleccionar um outro método para a resolução do novo problema. Dada a complexidade subjacente a esta nova situação, torna-se difícil encontrar um método que garanta a obtenção de uma solução óptima, pelo que se poderá aplicar um método heurístico, baseado, por exemplo, na regra de prioridade LPT (*Longest Processing Time*), em que os trabalhos são ordenados por ordem decrescente dos seus tempos de processamento, para darem início ao seu processamento num processador que fique livre mais cedo. Este procedimento permite, em geral, obter bons resultados, para este tipo de problemas, não garantindo, contudo, a obtenção de soluções óptimas.

Aplicando o procedimento heurístico referido (baseado na regra de despacho LPT) é possível obter os resultados que se apresentam a seguir na Tabela 6.13, que mais uma vez, indica os instantes de início e de fim de cada um dos trabalhos e o processador em que cada um deve ser processado.

Tabela 6.13 – Resultados do problema.

Trab. j	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
Máq. i	M4	M2	M1	M4	M1	M2	M2	M4	M2	M3	M4	M3	M1	M3	M3
Início	15	0	0	8	20	10	21	20	16	9	0	21	14	0	16
Fim	20	10	14	15	24	16	23	23	21	16	8	22	20	9	21

O tempo total dos trabalhos no sistema de processadores paralelos é de 24 U.T., portanto, ligeiramente superior ao valor obtido para o problema anterior, tal como era de esperar, dada a não permissão de interrupção no processamento dos trabalhos. Sequências de fabrico possíveis dos quinze trabalhos em cada um dos quatro processadores, M1 a M4, são as seguintes:

Processador M1: T3, T13, T5.

Processador M2: T2, T6, T9, T7.

Processador M3: T14, T10, T15, T12.

Processador M4: T11, T4, T1, T8.

A Figura 6.25 ilustra o correspondente diagrama de Gantt para o problema.

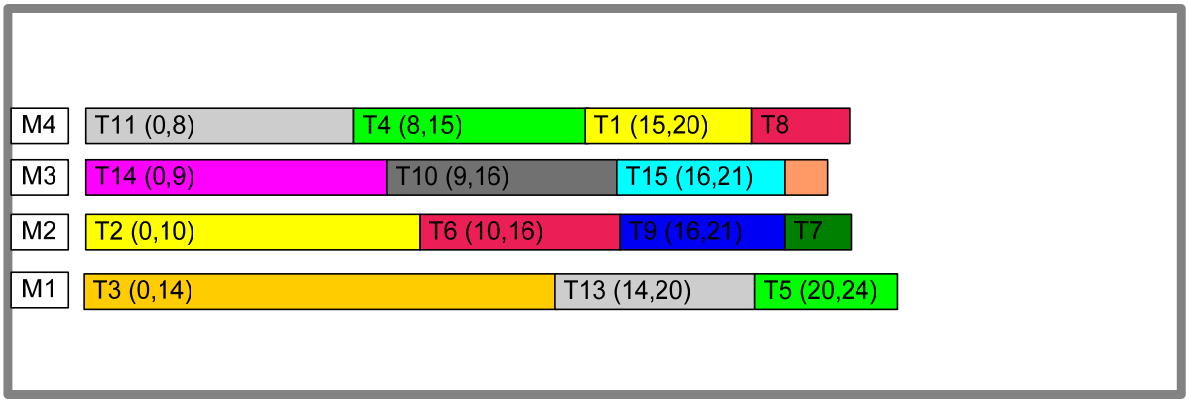

Figura 6.25 – Diagrama de Gantt: heurística para minimização de Cmax.

6.5 Considerações finais

Uma das funcionalidades do sistema é apresentar ao utilizador o programa de produção na forma pela qual o método o disponibiliza e ainda através de um processo adicional de representação baseado no digrama de Gantt. Outras formas de apresentação podem e devem ser explorados caso a caso, dependendo do interesse do utilizador. Esta funcionalidade embora relevante para o utilizador tem uma relevância de investigação menor, pelo que não foram exploradas alternativas adicionais de representação das soluções além do diagrama de Gantt.

Podemos antever ainda a possibilidade de confrontar soluções alternativas obtidas de métodos alternativos de resolução e eventualmente desenvolver mecanismos de manipulação das soluções com vista à obtenção de soluções híbridas mais vantajosas. Esta visão poderá justificar investigação futura por forma a, na alçada do utilizador, se chegar a soluções melhores sob o ponto de vista do utilizador.

Não foram equacionadas formas de escalonamento dinâmico no demonstrador desenvolvido. É natural que elas o sejam, em particular, a informação do estado do sistema em cada momento, deverá estar disponível e utilizável nos casos em que a assinatura dos métodos a possam requerer. Neste caso, o problema adicional que se põe é a conversão do formato dos dados recolhidos, relevantes, do estado do sistema numa forma compatível com assinatura dos métodos. Esta conversão requer programação adicional em XML que seria naturalmente dependente do utilizador.

Este capítulo permitiu ilustrar as funcionalidades essenciais do sistema web de apoio ao escalonamento da produção proposto (SWAEP), apresentando uma análise de diferentes problemas de escalonamento, resolvidos através de alguns métodos disponibilizados pelo sistema demonstrador desenvolvido de modo a testar a utilidade deste tipo de sistema, quando usado em diferentes cenários de produção.

Como se teve oportunidade de constatar pelos exemplos tratados, mesmo ligeiras variações nos problemas podem conduzir à necessidade de aplicação de métodos diferentes. Isto reforça a importância de um sistema desta natureza de modo a dar resposta aos diferentes cenários de escalonamento que possam surgir na realidade industrial, permitindo uma pesquisa alargada de diversos tipos de métodos de escalonamento da produção.

Pode concluir-se que o sistema proposto permite apoiar no processo de tomada de decisão no escalonamento da produção, permitindo a comparação de diferentes soluções, obtidas pela execução de métodos alternativos para um determinado problema e seleccionar a solução que se apresente mais adequada para a resolução de cada problema particular.

Embora a base de conhecimento de métodos inclua já, alguns métodos com implementações disponíveis para resolver alguns tipos de problemas, nomeadamente os tipos de problemas apresentados, o sistema "crescerá", na medida em que novas implementações de métodos sejam disponibilizadas, para a resolução de outros problemas.

7 Conclusões

7.1 Contribuições do trabalho

O trabalho aqui desenvolvido apresenta contribuições no domínio do escalonamento da produção em duas envolventes principais. Uma a um nível teórico, conceptual e outra ao nível prático da resolução de problemas através da colaboração de uma rede de agentes prestadores de serviços de escalonamento.

Ao nível conceptual contribui para uma ontologia de problemas de escalonamento da produção e conceitos relacionados, tais como métodos de resolução, implementações e soluções, providenciando um enquadramento comum para a compreensão e a partilha de conhecimento acerca destes conceitos.

Ao nível da resolução dos problemas de escalonamento, faz-se um estudo alargado sobre a forma da sua resolução com base num universo dos métodos de escalonamento da produção, distribuídos globalmente, adequados para problemas que ocorrem em diversos tipos de ambientes e sistemas de produção. Esta contribuição culmina com o desenvolvimento de um demonstrador de um sistema web de apoio ao escalonamento, SWAEP cuja arquitectura e funcionalidades são estudadas. O demonstrador, que pode ser visto como uma implementação possível do SWAEP, é usado para demonstrar o funcionamento do SWAEP e para avaliar a viabilidade das premissas e hipóteses subjacentes a este trabalho de investigação levantadas no primeiro capítulo. Entre elas é de realçar a pretensão inicial desta tese explorar e implementar o paradigma de escalonamento ubíquo colaborativo através da concepção e implementação de sistemas web de escalonamento da produção baseados em conhecimento distribuído globalmente e acedido via Internet e intranets. Esta filosofia difere das soluções de apoio ao escalonamento, ou existentes nas empresas ou actualmente disponíveis via Internet, tipicamente centralizadas e funcionando numa lógica cliente-servidor.

Assim, foi concebida uma arquitectura computacional do sistema SWAEP baseada no paradigma peer–to-peer de redes de computadores e numa base de conhecimento distribuído de escalonamento associada à rede. Este conhecimento foi estruturado com base

no trabalho teórico ontológico desenvolvido, do que se realça a identificação e relacionamento de uma diversidade de ambientes de produção e ainda a criação de uma nomenclatura capaz de representar uma gama muito alargada de problemas práticos.

O SWAEP foi modelado e implementado usando XML e tecnologias relacionadas, e as suas principais funcionalidades foram ilustradas, principalmente no capítulo 6.

Uma dificuldade importante encontrada neste trabalho foi a necessidade de organizar, relacionar e enquadrar os ambientes de escalonamento determinantes da especificidade de cada problema a resolver. A simples listagem de conceitos tradicionalmente abordados na literatura tornou-se insuficiente para implementar o processo de escalonamento proposto. Ainda que alguns conceitos apareçam relacionados na literatura, uma visão organizada, integradora e alargada deles e seu relacionamento não era claramente visível. A referência a ambientes de produção de uma forma limitada, com foco no *job-shop*, *flow-shop* puras e num conjunto restrito de critérios de optimização, naturalmente importantes para a investigação, não reflecte cabalmente as necessidades práticas de especificação de problemas e, portanto, é insuficiente para perspectivar a envolvente de aplicação industrial pretendida com este trabalho. É necessário considerar, por exemplo, vários arranjos no que concerne a composição de cada posto de trabalho e ao uso de mecanismos e recursos auxiliares, que vulgarmente não aparecem equacionados na investigação teórica mas que precisam de ser representados por caracterizarem os problemas práticos. De facto, por exemplo, os postos de trabalho podem ter vários processadores cada, que podem ser diferentes, podendo ainda ser providos de armazenagem local de dimensão variada e ferramentas auxiliares de processamento, em tipo e quantidade variável. Estes factores podem ser relevantes em muitos casos e terem impacto determinante na resolução de problemas, sendo por isso importantes para a selecção de métodos de escalonamento, e desta forma, condicionando e limitando a sua escolha.

Havendo nomenclaturas de representação dos problemas, não só identificando os seus ambientes de produção mas também os seus critérios de optimização e características particulares dos trabalhos e recursos, verificou-se no que eram limitadas face à complexidade e abrangência de especificação que se pretendia tratar e representar.

Assim, foram estudados tanto os ambientes de produção como outros factores caracterizadores dos problemas e, ainda, uma nomenclatura alargada de representação codificada de problemas.

Os ambientes de escalonamento identificados neste trabalho e a nomenclatura de representação de problemas proposta, juntamente permitem uma abrangência realista para especificar e representar a grande maioria dos problemas de escalonamento que se podem encontrar na prática.

Em particular, no capítulo 3, foram identificadas mais de quatro dezenas de ambientes de escalonamento, relacionados entre si pelo seu grau de generalidade e flexibilidade, desde situações muito gerais e flexíveis até situações muito restritivas e elementares. Cada caso deriva de outros mais complexos por parametrização dos factores caracterizadores dos ambientes de produção. Portanto, chegam-se aos ambientes mais simplificados a partir de restrições impostas a casos gerais e flexíveis numa perspectiva que se opõe à que normalmente é apresentada noutros trabalhos, como por exemplo em Baker (1974) e Brucker (1995) e numa lógica hierárquica, em que os sistemas mais simples se vêm como subclasses dos mais complexos.

Os factores de especificação dos ambientes de produção são identificados na nomenclatura pelas classes α, β e γ, como se apresentou no capítulo 4, bastante à semelhança da estrutura representativa de outras nomenclaturas, mas incluindo factores de representação adicionais, dentro das classes, com vista a representar a grande maioria dos problemas de escalonamento que se podem encontrar na prática. Desta forma é possível convenientemente caracterizar e formalmente descrever os problemas de escalonamento.

Um elemento fundamental na caracterização ou especificação de problemas de escalonamento é o ambiente de produção. Mostrou-se, no capítulo 4, que um ambiente de produção é identificado pela simbiose entre os componentes físicos do sistema de produção, principalmente processadores e os processos de fabrico dos trabalhos. Estes processos determinam as operações de transformação, a sequência pela qual podem ser realizadas e ainda os tipos e quantidade de processadores que podem ou devem ser utilizados em cada transformação. Quando nestas dimensões de configuração do ambiente de produção há flexibilidade, um problema prático pode ser encarado e resolvido na óptica de ambientes de produção diversos, podendo este configurar um problema de *general shop*, *job-shop*, *flow-shop*, flexível e/ ou de multi-processador e, desta forma, ser resolvido por diferentes métodos, uns mais abrangentes, outros mais restritivos e adequados apenas a ambientes igualmente restritivos. Esta flexibilidade realça o facto de que a clara identificação do ambiente de produção de um dado problema real de escalonamento pode

ser difícil e não é linear como se poderia pensar à partida. Dá no entanto alternativas ao utilizador na especificação do ambiente que se ajuste a métodos alternativos de resolução.

Conclui-se, portanto, que a identificação de cada problema de escalonamento depende e está intimamente relacionada com os meios físicos disponíveis, principais e auxiliares, e com os requisitos de transformação dos trabalhos podendo tomar diversas formas para o mesmo problema prático de produção, dependendo da imposição de restrições admissíveis face às possibilidades do sistema e flexibilidade de ordenação das sua operações de transformação.

Todos estes elementos são necessários e utilizados no SWAEP para identificação do problema a resolver e pesquisa de métodos adequados à sua resolução.

Esta pesquisa é realizada através de um processo de selecção de métodos adequados à resolução de cada problema de escalonamento, com base num esquema de associação entre problema especificado e método disponível. Este esquema baseia-se na correspondência entre características do problema, registadas através da utilização da nomenclatura de classificação dos problemas, e características do método. Estas são identificadas com base num modelo de descrição de assinaturas de métodos.

Os requisitos funcionais do SWAEP remeteram também para a necessidade de desenvolver um modelo para a especificação e identificação de assinaturas de métodos de escalonamento. A adopção deste modelo permite, para cada método, a especificação dos dados necessários e à sua execução na máquina onde está implementado, local ou remotamente, para obtenção de soluções.

A pesquisa de métodos faz-se na base de conhecimento distribuída, dispersa numa rede de agentes colaboradores, organizados na forma de uma rede P2P e formando uma comunidade virtual de escalonamento da produção (Varela, 2002a; 2002b).

A nomenclatura de classificação de problemas é importante à implementação do processo de selecção de métodos de escalonamento da produção dado permitir, não só, especificar as características genéricas dos problemas mas também características particulares que são importantes para a selecção de métodos para resolver os correspondentes problemas.

Para validar e testar o SWAEP, desenvolveu-se um sistema web demonstrador, implementando tanto a arquitectura concebida como as funcionalidades requeridas, com base na tecnologia XML. Esta tecnologia permite que de uma forma adequada e

relativamente simples se possa especificar e usar os métodos de escalonamento distribuídos na rede P2P subjacente ao sistema.

Dos resultados obtidos e usando o demonstrador, podemos concluir que praticamente todas as funcionalidades planeadas para implementar o SWAEP foram testadas e mecanismos necessários desenvolvidos tendo-se demonstrado a viabilidade do uso do paradigma ubíquo colaborativo de escalonamento, baseado em serviços web centralmente equacionado nesta tese. No capítulo 6 foi exemplificado todo processo, desde a especificação de problemas à identificação e invocação dos métodos para a sua resolução. Para tal foi necessário implementar alguns métodos e usá-los para demonstrar este processo de escalonamento na estrutura distribuída de escalonamento.

Um sistema destes claramente configura transferência de tecnologia para o sector empresarial, através da partilha de recursos, com particular destaque para um, que é essencial, que é o conhecimento, relativo aos problemas de escalonamento e à forma de os resolver.

7.2 Trabalho futuro

Como se refere acima, a maioria dos mecanismos e funcionalidades requeridas a qualquer SWAEP seguindo a abordagem ao escalonamento adoptada neste trabalho, podem ser implementadas com sucesso, como ficou demonstrado através do sistema demonstrador desenvolvido. No entanto, há alguns aspectos complementares importantes, que merecem a atenção de trabalho futuro.

Em especial é de referir que como está, a selecção de métodos é dependente do utilizador e não de mecanismos automáticos ou semi-automáticos de selecção qualitativa de métodos, baseada na qualidade das soluções, na eficiência e/ ou no custo associado à sua utilização. Estes requisitos são importantes num sistema de escalonamento, do tipo do desenvolvido e proposto, pelo que devem naturalmente ser equacionados, em trabalho futuro, com vista à selecção qualitativa de métodos. Assim, perante a existência, na base de conhecimento distribuída, de métodos alternativos para a resolução de um problema, deverá ser possível seleccionar o método mais adequado com base em critérios de selecção que tomam em conta, separadamente ou conjuntamente, aquelas e/ ou outras qualidades dos métodos.

Claramente que se antevê a necessidade de realização prévia de outra tarefa associada e muito importante que é a caracterização qualitativa dos métodos, sem o que a viabilidade de uma selecção apropriada dos métodos poderia ficar comprometida, a menos da grande

perícia do utilizador. Tal tarefa é muito importante, porquanto a selecção qualitativa dos métodos só é viável perante a existência de dados e seu tratamento objectivo que permita a caracterização dos métodos na base das qualidades ou varáveis referidas. Isto não é um trabalho fácil podendo requerer investigação significativa para ser convenientemente realizado.

Podemos antever de três caminhos, entre outros, para chegar a resultados aceitáveis neste contexto:

Avaliar cada métodos individualmente, à medida que é utilizado, em relação a uma amostra representativa de problemas de escalonamento a resolver. Para isto a própria base de conhecimento distribuída (BCD) poderia desempenhar aqui um papel importante. De facto , a BCD contém, problemas que são especificados pelo módulo de especificação de problemas do SWAEP. Portanto, uma amostra de problemas pode ser seleccionada do histórico repositório dos problemas já especificados. Este repositório pode ser enriquecido com a recolha de dados sobre os problemas da amostra que tenham sido resolvidos. Claramente isto obrigaria a uma recolha de dados, para além dos programas de produção oferecido pelos métodos. Para tal novas funcionalidade teriam de ser adicionadas ao SWAEP para que esse dado relativamente às variáveis, qualidade, eficiência e custo pudessem ser recolhidos em cada execução dos métodos como resultados adicionais ao programa de produção.

Recolher informação disponível publicada ou prestada pelo agente de prestação de serviço de escalonamento. Uma tarefa bem mais fácil que a anterior, mas provavelmente menos fiável.

Usar métodos formais de avaliação da qualidade do métodos baseado nas variáveis referidas através de um modelo de avaliação de métodos que permita perspectivar as variáveis referidas, de forma independente e integrada. Neste caso a importância de cada critério e avaliação individual teria de ser equacionada, o que poderia ser dependente de utilizador para utilizador.

Podemos antever ainda a possibilidade de confrontar soluções alternativas obtidas de métodos alternativos de resolução e eventualmente desenvolver mecanismos de manipulação das soluções com vista à obtenção de soluções híbridas mais vantajosas. Esta visão poderá justificar investigação futura por forma a, na alçada do utilizador, se chegar a soluções melhores compatíveis com critério de optimização.

Não foram equacionadas formas de escalonamento dinâmico no demonstrador desenvolvido. É natural que, elas o sejam, em particular, a informação do estado do sistema em cada momento, deverá estar disponível e utilizável nos casos em que a assinatura dos métodos a possam requerer. Neste caso, o problema adicional que se põe é a conversão do formato dos dados recolhidos, relevantes, do estado do sistema numa forma compatível com assinatura dos métodos especificada em XML para cada método. Esta conversão requer programação adicional em XML. Este tema poderá justificar, perante implementações do SAWEP por utilizadores um estudo completar para possibilitar o escalonamento dinâmico ou reescalonamento.

8 Referências

Abiteboul, S., et al, *Data on the web: From relations to semistructured data and XML* (USA: Morgan Kaufmann Publishers, 2000).

Agnetis, A., Alfieri, A., Brandimarte, P., "Join Job/ Tool Scheduling in a Flexible Manufacturing Cell with No-Board Tool Magazine", Computer Integrated Manufacturing Systems, Vol. 10, Nº 1, 61-68, 1997.

Almeida, A. Ramos C., Silva, S. C. Product Oriented Scheduling through Job Scheduling Patterns, IEPM'03 International Conference on Industrial Engineering and Production Management, Porto, Portugal, May 2003.

Amar, A. D., Xiao, B., "Scheduling on a Bottleneck Station: A Comprehensive Cost Model and Heuristic Algorithms", International Jornal of Production Research, Vol. 35, Nº 4, 1011-1030, 1997.

Aparício J. N., et al, *Applications development with XML* (Portugal: New University of Lisbon, 2001).

Aparício JN et al, Applications development with XML. Portugal: New University of Lisbon, 2001.

Artiba, A., Elmaghraby, S. E., *The planning and scheduling of production systems* (UK: Chapman & Hall, 1997).

Arts, E., Lenstra, J. K., 1997. *Local Search in Combinatorial Optimization*, John Wiely & Sons.

Atlan, Bonnet, Naillon, "Learning Distributed Reactive Strategies by Genetic Programming for the General Job Shop Problem", Proceedings of the 7th Annual Florida Artificial Intelligence Research Symposium, Pensacola, Florida, USA, IEEE Press, Vol. 11, 1994.

Atlan, Bonnet, Naillon, "Learning Distributed Reactive Strategies by Genetic Programming for the General Job Shop Problem", Proceedings of the 7th Annual Florida Artificial

Intelligence Research Symposium, Pensacola, Florida, USA, IEEE Press, Vol. 11, 1994.

Azevedo, M., "Teses, Relatórios e Trabalhos Escolares – sugestões para a sua elaboração", Faculdade de Ciências da Universidade de Lisboa, Departamento de Educação, 1994.

Azizoglu, M., Webster, S., "Scheduling Job Families About an Unrestricted Common Due Date on a Single Machine", International Jornal of Production Research, Vol. 35, Nº 5, 1321-1330, 1997.

Badiru, A., B., "Expert Systems Applications in Engineering and Manufacturing", Prentice-Hall, Inc., 1992.

Baer, P. P., "Minerva Gui: The Knowledge Base Editor", KBS Research Group, Beckman Institute, University at Urbana - Champaign,1996.

(http://www-kbs.ai.uiuc.edu/home-main.html)

Baker, K., Introduction to Sequencing and Scheduling. John Wiely & Sons, 1974.

Barman, S., "Simple Priority Rule Combination: An Approach to Improve both Flow Time and Tardiness", International Jornal of Production Research, Vol. 35, Nº 10, 2857-2870, 1997.

Battiti, R., "Reactive Search Heuristics: Toward Self-Tuning", Dipartimento di Matematica, Università di Trento, Italy, 1996.

Bauer, A., Bowden, R., Browne, J., Duggan, J., Lyons, G., "Shop Floor Control Systems – From Design to Implementation", 1991.

Beaud, M., "Arte da Tese", Bertrand, 1996.

Bedworth, D., Baile, J. E., "Introduction to Production Control Systems", John Wiley & Sons, 1987.

Beynon-Davies, P., "Expert Database Systems", McGraw-Hill, 1991.

Bhaskar, K., Spinivasan, G., "Static and Dynamic Operator Allocation Problem in Cellular Manufacturing Systems", International Jornal of Production Research, Vol. 35, Nº 12, 3467-3481. 1997.

Blazewicz, J., Ecker, K., H., Pesch, E., Schmidt, G., Weglarz, J. "Scheduling Computer and Manufacturing Processes", Springer-Verlag, 1996.

Blurock, E. S., "Course: Knowledge Based Engineering", Research Institute for Symbolic Computation (Risc-Linz), Johannes Kepler University, Linz, Austria,1997.

(http://info.uni-linz.ac.at:/people/blurock/courses/expert/expert.html)

Botelho, F., Moura, A., Simões, F., A., Pinto, M., L., C., "Rudimentos de Programação numa Linguagem de Alto Nível – Introdução às Tecnologias de Informação (8)", Edições ASA, 1995.

Bourgade, V., Oulamara, A., "Scheduling the Arrivals in a No-wait Flowshop with Fuzzy Processing Times", LMCM, 1998.

Bowyer, C., "Beginning JavaScript", Wrox Press Ltd., 2001.

Brownell, "Conformance Testing para Processadores de XML", 1999.

Brucker, P., "Interactive Classification",1998.

(http://www.mathematik.uni-osnabrueck.de/research/or/class/)

Brucker, P., "Scheduling Algorithms" Springer-Verlag, 1995.

Bryant, N., "Managing Expert Systems", John Wiley & Sons, 1988.

Burbidge, J., L., "IFIP Blossary of Terms Used in Production Control", IFIP, 1987.

Camarinha-Matos, L. M., Steiger-Garção, A., "Robotic Cell Programming: A Knowledge-Based Approach", 8th IASTED/AFCET Int Symp on Robotics and Artificial Intelligence, Toulouse 18-20 Junho 1986. (http://www.uninova.pt/CRI/MEMBERS/cam/pub90.htm)

Camarinha-Matos, L. M.; Afsarmanesh, H., Design of a Virtual Community Infrastructure for Elderly Care, In: *Collaborative Business Ecosystems and Virtual Enterprises, PRO-VE'02*, Sesimbra, Portugal. Kluwer Academic Publishers, 2002, pp. 439-450.

Carlson, D. (2001). *Modeling XML Applications with UML*. Addison-Wesley, USA.

Carmo-Silva, S., Alves , A.C., Costa M., (2005), "A Computer Aided Design System for Product Oriented Manufacturing Systems Reconfiguration", Proceedings of the Intelligent Production Machines and Systems Conference, Kluwer.

Carriço, J. A. S., António João Chambel da Silva Carriço, "Java – A Linguagem da Internet", Centro de Tecnologias de Informação Lda., 1997.

Ceponkus, A., F. Hoodbhoy, *Applied XML* (USA: Wiley Computer Publishing, 1999).

Chrétienne P., et al, *Scheduling theory and its applications* (England: John Wiley & Sons Inc., 1995).

Claβen, I., Hartmut Ehrig, Dietmar Wolz, "Algebraic Specification Techniques and Tools for Software Development", World Scientific, 1993.

Coelho, P., "A Nova Linguagem da Web", FCA – Editora de Informática Lda., 2000.

Conway, R. W., Maxwell, W. L., Miller, L. W., Theory of Scheduling. 1967, England: Addison-Wesley Publishing Company, Inc.

Dagli, C. H. (1994). Artificial Neural Networks for Intelligent Manufacturing. Chapmann & Hall.

Dagli, Sittisathanchai, "Genetic Neuro-Scheduler for Job-Shop Scheduling", Computers and Industrial Engineering, Vol. 25., Nº 4, 267-270, 1993.

Dennis Merritt, "Building Expert Systems in Prolog", Springer-Verlag, 1989.

Dhaliwal, J., S., Benbasat, I., "The Use and Effects of Knowledge-Based System Explanations: Theoretical Foundations and a Framework for Empirical Evaluation", Information Systems Research, Vol. 7., Nº 3, 342-362, 1997.

Dirk, Mattfeld, "Evolutionary Search and the Job-Shop. Investigations on Genetic Algorithms for Production Scheduling", Physica-Verlag, 1996.

Edward S. Blurock, "Course: Knowledge Based Engineering", Research Institute for Symbolic Computation, 1998.

El-Najdawi, M. K., "Multi-cyclic Flow Shop Scheduling: An Application in Multi-Stage, Multi-Product Production Processes" International Jornal of Production Research, Vol. 35, Nº 12, 3323-3332, 1997.

Eloranta, E., "Advances in Production Management Systems", IFIP, 1991.

Evans, T., "HTML-Simples e Rápido", Makron Books, 1996.

Fang, Ross, Corne, "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems", Proceedings of the Fifth International Conference on Genetic Algorithms, (:) 375-382.

Fichera, Grasso, Lombardo, Lovalvo "Genetic Algorithms Efficiency in Flow-Shop Scheduling", Applications of Artificial Intelligence in Engineering (:), 261-270.

Fox. R. E. 1983. "OPT – An Answer for America. Part IV – Leapfrogging the Japanese" Inventories and Production Management.

Frederick Hayes-Roth, Donald A. Waterman, Douglas B. Lenat, "Building Expert Systems", Addison-Sesley Publishing Company, Inc., 1983.

French, S. (1982). Sequencing and Scheduling – An Introduction to Mathematics of the Job-Shop. John Wiley and Sons, Inc.

Fujimoto, Yasuda, Tanigawa, Iwahashi, "Applications of Genetic Algorithm and Simulation to Dispatching Rule-Based FMS Scheduling", IEEE International Conference on Robotics and Automation (:), 190-195.

Gallant, S., I., "Newral Network Learning and Expert Systems", The Mit Press, 1993.

Giarratano, C. J., "CLIPS Reference Manuals and Users's Guide", Ph. D., NASA, 1997.

(http://www.ghg.net/clips/download/documentation/.)

Godfrey C. Onwubolu, M. Multingi, *Genetic Algorithm for Minimizing Tardiness in Flow-shop Scheduling*. Production Planning and Control, 1999. Vol. 10(5): p. 462-471.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc.

Graham, R. L., et al. (1979). Optimisation and Approximation in Deterministic Sequencing and Scheduling: A survey. Annals of Discrete Mathematics.

Gu, P., Norrie, D., H., "Intelligent Manufacturing Planning", Chapman&Hall, 1995.

Gupta, J. N. D., "A Flowshop Scheduling Problem with Two Operations per Job", International Jornal of Production Research, Vol. 35, Nº 8, 2309-2325. 1997.

Hannu Kangassalo, Hannu Jaakkola, Setsuo Ohsuga, Bent Wangler, "Information Modelling and Knowledge Bases VI, IOS Press, 1995.

Harbison,-Briggs, K., "Knowledge Acquisition – Principles and Guidelines", Prentice-Hall International Editions, 1989.

Harper, F., *XML standards and tools* (USA: eXcelon Corporation, 2001).

Herwijnen, E. "Practical SGML", Kluwer Academic Publishers, 1990.

Herzner, J, Kubiska, M., "Supplement to Expert Systems Tutorial: Rule Basing", 1992.

(http://www.rpi.edu/dept/chem-eng/Biotech-Environ/EXPERT/expert1.html)

Hitomi, K., "Manufacturing Systems Engineering", Taylor & Francis, 1996.

Holsapple, Jacob, Pakath, Zaveri, "A Genetics-Based Hybrid Scheduler for Generating Static Schedules in Flexible Manufacturing Contexts", IEEE Transactions on Systems Man and Cybernetics, Vol. 23., Nº 4, 953-972, 1993.

Holzner, S., "JavaScript Complete", McGraw-Hill, 2001.

Hong, S. J., "An Iterative Approach for Generating Minimal Rules from Examples", IEEE Transactions on Knowledge and Data Engineering, Vol. 9, Nº 5, 710-716, 1997.

http://www.componentsonline.com/Middleware/Explained/default.html

Hunter, D., "Beginning XML", Wrox Press Ltd., 2001.

Ignall, E., Schrage, L. (1965). Application of the Branch-and-Bound Technique to Some Flow-Shop Problems. *Operations Research*, 13, 400-412.

Ishibuchi, Yamamoto, Murata, Tanaka, "Genetic Algorithms and Neighborhood Search Algorithms for Fuzzy Flow-Shop Scheduling Problems", Fuzzy Sets and Systems, Vol. 67., Nº 1, 81-100, 1994.

Jacek B., Klaus H. Ecker, Erwin P., Gunter Schmidt, Jan W., "Scheduling Computer and Manufacturing Processes", Springer-Verlag, 1996.

Jordan, C., "Batching and Scheduling", Springer-Verlag, 1996.

Julie Wallin Kaewert, John M. Frost, "Developing Expert Systems for Manufacturing – Case Study Approach", McGraw-Hill, 1990.

Kadipasaoglu, S. N., Xiang, W., Khumawala, B. M., "A Comparison of Sequencing Rules in Static and Dynamic Hibrid Flow Systems", International Jornal of Production Research, Vol. 35, Nº 5, 1359-1384. 1997.

Kadipasaoglu, S. N., Xiang, W., Khumawala, B. M., "A Note of Scheduling Hybrid Flow Systems", International Jornal of Production Research, Vol. 35, Nº 5, 1491-1494. 1997.

Kaskavelis, C. A., Caramanis M. C., (1996). Application of Lagrangian Relaxation Scheduling Based Algorithms to Semicondutor Testing Facility, *Technical Paper of the Department of Manufacturing Engineering*, Boston University, Boston.

Keller, R., "Tecnologia de Sistemas Especialistas", Makron Books Ltda., 1991.

Kelton, W., D., Sadowski, R., P., Sadowski, D., A., "Simulation with Arena", McGraw-Hill, 1998.

King, J., R., "Production Planning and Control", Programon International Library, 1975.

Kusiak, A., "Intelligent Manufacturing Systems", Prentice-Hall Inc., 1990.

Laurent, S., et al. (2001). *Programming Web Services with XML-RPC*. O'Reilly & Associates, Inc.

Lee, "Genetic Algorithms for Single-Machine Job Scheduling with Common Due-date and Symmetrical Penalties", Jornal of the Operations Research Society of Japan, Vol. 37, Nº 2, 83-95, 1994.

Lee, C.Y., Piramuthu, S., Tsai, J.K., *Job Shop Scheduling with Genetic Algorithm and Machine Learning*. International Journal of Production Research, 1997. Vol. 35(4): p. 1171-1191.

Lee, Choi, "A Genetic Algorithm for Job Sequencing Problems with Distinct Due-dates and General Early-Tardy Penalty Weights", Computers and Operations Research, Vol. 22, Nº 8, 857-869, 1995.

Lee, Sikora, "Joint Lot-Sizing and Sequencing with Genetic Algorithms for Scheduling – Envolving the Chromosome Structure", Proceedings of the Fifth International Conference on Genetic Algorithms, 383-389, 1993.

Levine, R., I., Drang, D., E., Edelson, B., "Inteligência Artificial e Sistemas Especialistas – Aplicações e Exemplos Práticos", McGraw-Hill, Inc., 1986.

Liang, Mannion, "Scheduling of a Flexible Assembly System using Genetic Algorithms", Imeche Conference Transactions, Nº 3, 173-178, 1995.

Liebowitz, J., "The Dynamics of Decision Support Systems and Expert Systems", The Dryden Press, 1990.

Lowe, D., Hall, W. "Hypermedia & the Web", John Wiley & Sons, Ltd., 1999.

Luh, P. B., Wang, J. H., Wang, J. L., Tomastik, R. N., "Near-Optimal Scheduling of Manufacturing Systems with Presence of Batch Machines and Setup Requirements", The CIRP, Vol. 46, 1997.

Mahmoodi, F., Martin, G. E., "A New Shop-based and Predictive Scheduling Heuristic for

Cellular Manufacturing", International Jornal of Production Research, Vol. 35, Nº 2, 313-326. 1997.

Maimon, O.Z., Bhaha, D., *A Genetic Algorithm Approach to Scheduling PCBs on a Single Machine*. International Journal of Production Research, 1997. Vol. 34(2): p. 1162-1170.

Mao, Wu, "Genetic Algorithm and the Application for Job-Shop Group Scheduling", Proceedings of the Society of Photo-Optical Instrumentation Engineers (Spie), Vol. 2620, (:), 103-108, 1995.

Marchal, B., "XML by Example", QUE Corporation, 2000.

Mau, R., Keyes, J., "Handbook of Expert Systems in Manufacturing", McGraw-Hill, Inc.,1991.

Merritt, D., "Building Espert Systems in Prolog", Springer-Verlag, 1989.

Monden, Y. "Toyota production System" Industrial Engineering and Management Press, Institute of Industrial Engineers, 1983.

Morton, T., D. W. Pentico, *Heuristic scheduling systems* (USA: John Wiley & Sons Inc., 1993).

Navarro, A., White, C. Burman, L. "Mastering XML", SYBEX Inc., 2000.

Neves, M., Belo, O., "Sistemas Periciais", Publicação Interna do Departamento de Informática, Universidade do Minho, Portugal, 1991.

Nurmela, K., J., Ostergard, P., R., J., "Constructing Covering Designs by Simulated Annealing", Helsinki University of Technology, Finland, 1993.

Oliveira, J., N., "Métodos Formais de Programação", Publicação Interna do Departamento de Informática, Universidade do Minho, Portugal, 1997.

Osman, I. H., Kelly, J. P., 1996. Meta-Heuristics: Theory and Aplications, Kluwer Academic Publishers.

Papazoglou, M.P.; Krämer, B.J.; Yang, J.; Leveraging Web-Services and Peer-to-Peer Networks, In: *Proceedings of Advanced Information Systems Engineering, 15th International Conference, CaiSE*, Klagenfurt, Austria. June 16-18, 2003, pp. 485-501.

Pierreval, H., Mebarki, N., "Dynamic Selection of Dispatching Rules for Manufacturing Systems", " International Jornal of Production Research, Vol. 35, Nº 6, 1575-1591, 1997.

Pinedo, M., *Scheduling theory, algorithms and systems* (USA: Prentice-Hall Inc., 2002).

Pires, J. N., "Automação Industrial", ETEP – edições técnicas e profissionais, Lda. 2002.

Powell, T. A., David L. Jones, Dominique C. Cutts, Beyound Web Page Design", Prentice Hall Inc., 1998.

Puppe, F., "Systematic Introduction to Expert Systems", Springer-Verlag, 1993.

Rabelo, R., Camarinha-Matos, L. M., "Control and Dynamic Scheduling in Virtual Organization of Production Resources", IFIP Int. Conf. On Evaluation of Production Management Methods, 21-24 Março, Porto Alegre, Brasil, 1994. (http://www.uninova.pt/CRI/MEMBERS/cam/pub94.htm)

Rabelo, R.J.; Camarinha-Matos, L.M.; Afsarmanesh, H., "Multi-agent perspectives to agile scheduling. In: *Intelligent Systems for Manufacturing*", Kluwer Academic Publishers, pp.51-66, Boston, 1998.

Ramalho, J. C., Henriques, P.: XML & XSL da Teoria à Prática. FCA, Editora de Informática Lda., Portugal (2002).

Randhawa, S. U., Kuo, C. H., "Evaluating Scheduling Heuristics for Non-Identical Parallel Processors" International Jornal of Production Research, Vol. 35, Nº 4, 969-981, 1997.

Raymond, McLeod, JR., "Management Information Systems", Prentice-Hall, Inc., 1995.

Ribeiro, R.A., "Apontamentos da disciplina de Lógica Difusa", Mestrado em Inteligência Artificial Aplicada, 1999 (Publicação Interna da FCT/UNL).

Rich, E., Knight, K., "Artificial Intelligence", McGraw-Hill, Inc., 1991.

Richard Forsyth, "Expert Systems – Principles and Case Studies", Chapman and Hall, 1984.

Russel, S., Norvig, P., "Artificial Intelligence – A Modern Approach", Prentice-Hall, Inc., 1995.

Salgueiral, A., Neves, J., Miranda, A., "Dicionário de Informática", Controljornal Editora, Lda., 1998.

Sharadapriyadarshini, B., Rajendran, C., "Formulations and Heuristics for Scheduling in Buffer- Constrained Flowshop and Flowline-based Manufacturing Cell with Different Buffer-space Requirements for Job: Part2", International Jornal of Production Research,

Vol. 35, Nº 1, 101-122, 1997.

Silva, S. C., "Organização e Gestão da Produção", Publicação Interna do Departamento de Produção e Sistemas, Universidade do Minho, 1993.

Silva, S. C., Alves, A. C., "Diferentes Perspectivas de Sistemas de Produção Orientados ao Produto", Publicação Interna do Departamento de Produção e Sistemas, Universidade do Minho, Portugal, 1997.

Silva, S. C., Alves, A. C., Perspectivas de sistemas de produção orientados ao produto, *TeamWork'2001 Conference*, Lisbon, Portugal, 2001, 1-19.

Silva, S. C., Varela, M. L. R., "Classificação e Descrição Formal de Sistemas de Produção Orientados ao Produto na Óptica do Escalonamento da Produção", Publicação Interna do Departamento de Produção e Sistemas, Universidade do Minho, Portugal, 1999.

Singh, N., Rajamani, D., "Celluar Manufacturing Systems – Design, Planning and Control", Chapman & Hall, 1996.

Smart, J., "WxCLIPS 1.64 (WIN 32) – An Environment for Developing KBS Applications with Graghical User Interfaces", NASA, 1996.

(http://web.ukonline.co.uk/julian.smart/wxclips/wxclips2.html)

Smart, J., "WxWindows Dialog Editor 1.6", NASA, 1996.

(ftp://www.remstar.com/pub/wxwin/wxclips/dialoged_win95/setup32.zip)

Soon, T. H., Souza, R., "Intelligent Simulation-Based Scheduling of Workcells: An Approach", Integrated Manufacturing Systems, Vol. 8, Nº 1, 6-23, 1997.

Sousa, I. D., "Negócios & Internet", FCA – Editora de Informática, Lda., 1997.

Sridhar, Rajendram, "A Genetic Algorithm for Family and Job Scheduling in a Flowline-based Manufacturing Cell", Computers and Industrial Engineering, Vol. 27., Nº 4, 469-472, 1994.

Stevens, J. W., Gemmill, D. D., "Scheduling a Two-Machine Flowshop with Travel Times to Minimize Maximum Lateness", International Jornal of Production Research, Vol. 35, Nº 1, 1-15. 1997.

Stross, C., "The Web Architect's Handbook", Addison Wesly, 1996.

Suresh, N. C., Kay, J. M.,"Group Technology and Cellular Manufacturing – State-of-the Art Synthesis of Research and Practice", Kluwer Academic Publishers, 1998.

Terziyan, V.; Zharko, A.; Semantic Web and Peer-to-Peer: Integration and Interoperability in Industry, Industrial Ontologies Group, MIT Department, University of Jyvaskyla, Finland (http://www.cs.jyu.fi/ai/vagan/papers.html).

Thomas E. Vollmann, William L. Berry, D. Clay Whybark, "Manufacturing Planning and Control Systems", IRWIN, 1997.

Van Der Zee, D. J., Van Harten, A., Schuur, P. C., "Dynamic Job Assignment Heuristics for Multi-Server Batch Operations – A Cost Based Approach", International Jornal of Production Research, Vol. 35, Nº 11, 3063-3093. 1997.

Varela, L. R., Aparício, J. N., Silva, S. C., (2002). Scheduling Problems Modeling with XML. In: *International Meeting for Research in Logistics* (Carvalho, J. C., et al. eds.), pp.897-909, International Meeting for Research in Logistics, Inc., Lisbon, Portugal.

Varela, L. R., Aparício, J. N., Silva, S. C., "An XML Knowledge Base System for Scheduling Problems", Springer-Verlag in the Lecture Notes in Computer Science series - Proceedings of the Innovative Internet Computing System Conference (I2CS), 20-22 June 2002, Kuhlungsborn, Germany.

Varela, M. L. R., *Automatic scheduling algorithms selection*, (Portugal: Msc. Thesis, University of Minho, 1999).

Varela, M. L. R.; Aparício, J. N.; Silva, S. C., Developing a Web Scheduling System Based on XML Modeling, In: Knowledge and Technology Integration in Product and Services – Balancing Knowledge and Technology in Product and Service Life Cycle, BASYS'02, Cancun, Mexico. Kluwer Academic Publishers, 2002, pp. 61-70.

Varela, M.L.R., Algoritmos Evolutivos Aplicados ao Escalonamento da Produção, Technical Report, 2000. N.º 5, University of Minho, CESP.

Vincenzo Ambriola, Genoveffa Tortora, "Advances in Software Engineering and Knowledge Engineering", World Scientific, 1993.

Womack JP. and Jones DT. Lean Thinking. Siman & Schuster, New York, USA,1996.

Wong, "Combined Genetic Algorithm for Generator Scheduling Techniques- Simulated Annealing Fuzzy Set Approach to Short-term Generation Scheduling", IEEE Transactions

on Power Systems, Vol 11, Nº1, 128-135, 1996.

Wu, J.; Distributed System Design. New York: CRC Press, 1999.

Yamada, Nakano, "A Genetic Algorithm Applicable to Large Scale Job-Shop Problems", Springer-Verlag, 1992.

Yen, B. et al. Internet Scheduling Environment with Market-Driven Agents. IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans, Vol. 34, No. 2, March 2004.

Zhou, M. (1995). *Petri Nets in Flexible and Agile Automation*. Kluwer Academic Publishers, London.