Sistema de Planeamento Fino da Produção com Sequenciamento de Lotes e Agendamento de Máquinas

Dissertação de Mestrado de

António Jorge da Silva Trindade Duarte

Realizada sob a orientação de

Prof. Doutor José Manuel Vasconcelos Valério de Carvalho

Resumo

Este trabalho é dedicado ao planeamento operacional, especificamente ao planeamento da produção em máquinas paralelas. O problema estudado é completamente preemptivo: qualquer tarefa pode ser interrompida e retomada mais tarde, na mesma máquina ou em outra qualquer, e pode haver multiprocessamento. Na abordagem é utilizada uma modificação do algoritmo de Horn, de modo a suportar uma matriz de adequação máquina/tarefa. Como existem tempos de preparação, a solução é melhorada recorrendo à heurística PDEDD e a uma heurística de trocas desenvolvida. Foi desenvolvida uma implementação computacional com a qual foram realizados testes à eficácia das heurísticas na redução de preempções desnecessárias. Esses testes são apresentados e os seus resultados discutidos.

Detailed Production Planning System with Lot Sequencing and Machine Scheduling

Master in Science dissertation of

António Jorge da Silva Trindade Duarte

Under the supervision of

José Manuel Vasconcelos Valério de Carvalho (PhD)

Abstract

This work is about operational planning, specifically about operational planning on parallel machines. The problem under study is fully preemptive: any task may be interrupted and resumed later, on the same machine or on another machine, and multiprocessing can occur. The approach uses a modification of Horn's algorithm, in order to support a machine/task adequacy matrix. Because there are setup times, the solution is improved by the PDEDD heuristic and by a developed swap heuristic. We developed a computer code, which was used to run some tests on the heuristics effectiveness in reducing the number of unnecessary preemptions. These tests are presented and their results are discussed.

Sistema de Planeamento Fino da Produção com Sequenciamento de Lotes e Agendamento de Máquinas

Dissertação de Mestrado de

António Jorge da Silva Trindade Duarte

Realizada sob a orientação de

Prof. Doutor José Manuel Vasconcelos Valério de Carvalho

Errata

Página 23 Linha 10 Onde se lê "Este matriz ..." deve ler-se "Esta matriz"

Página 35 Linha 3 Onde se lê "... esta restrições ..." deve ler-se "... esta restrição ..."

Página 43 Linha 6 Onde se lê "... o critério do tamanho ..." deve ler-se "... o critério da minimização do tamanho ..."

Página 66 Linha –4 Onde se lê "... paras as tarefas ..." deve ler-se "... para as tarefas ..."

Página 100 Linha 2 Onde se lê "... produz bom resultados." deve ler-se "... produz bons resultados."

AGRADECIMENTOS

À Fiorima, nas pessoas do Dr. Machado Rodrigues e do Eng. José Rodrigues, pela colaboração, apoio e total disponibilidade com que sempre nos receberam.

À Efacec, na pessoa do Eng. Cunha Rodrigues, pela preciosa ajuda nos contactos com a Fiorima.

Ao Dr. Valério de Carvalho, pelos conselhos sábios e oportunos, um orientador em todos os sentidos da palavra.

Trabalho co-financiado pelo FSE.

MINISTÉRIO DA EDUCAÇÃO prodep PROGRAMA DE DESENVOLVIMENTO EDUCATIVO PARA PORTUGAL	COMUNIDADE EUROPEIA Fundo Social Europeu	

RESUMO

Este trabalho é dedicado ao planeamento operacional, especificamente ao planeamento da produção em máquinas paralelas. O problema estudado é completamente preemptivo: qualquer tarefa pode ser interrompida e retomada mais tarde, na mesma máquina ou em outra qualquer, e pode haver multiprocessamento. Na abordagem é utilizada uma modificação do algoritmo de Horn, de modo a suportar uma matriz de adequação máquina/tarefa. Como existem tempos de preparação, a solução é melhorada recorrendo à heurística PDEDD e a uma heurística de trocas desenvolvida. Foi desenvolvida uma implementação computacional com a qual foram realizados testes à eficácia das heurísticas na redução de preempções desnecessárias. Esses testes são apresentados e os seus resultados discutidos.

ABSTRACT

This work is about operational planning, specifically about operational on parallel machines. The problem under study is fully preemptive: any task be interrupted and resumed later, on the same machine or on another and multiprocessing can occur. The approach uses a modification of Horn's algorithm, in order to support a machine/task adequacy matrix. Because setup times, the solution is improved by the PDEDD heuristic and by a developed swap heuristic. We developed a computer code, which was used some tests on the heuristics effectiveness in reducing the number of preemptions. These tests are presented and their results are discussed.

ÍNDICE GERAL

Capítulo 1
INTRODUÇÃO

O tema deste trabalho situa-se na área do planeamento da produção. Segundo a divisão clássica do planeamento da produção, em planeamento estratégico (longo prazo, 3-5 anos), planeamento táctico (médio prazo, 1 ano) e planeamento operacional (curto prazo, dias/semanas/meses), o tema deste trabalho insere-se claramente no âmbito do planeamento operacional.

A motivação para este trabalho surgiu na sequência de contactos havidos entre o autor e uma empresa do sector têxtil, onde o trabalho foi desenvolvido. Como a empresa estava numa fase de reestruturação do seu sistema de informação e de apoio à decisão, surgiu a oportunidade e a necessidade de fazer melhorias na área do planeamento da produção. Um dos aspectos que deveria ser revisto era o problema do agendamento da produção num conjunto de máquinas paralelas em determinada parte do processo produtivo.

Num sector onde os baixos tempos de produção são absolutamente essenciais, o planeamento da produção é crítico. Por outro lado, a forte concorrência, tanto a nível nacional, como a nível internacional, determinam a necessidade de melhorias constantes e de aumentos de eficiência. O planeamento da produção torna-se um factor determinante para o sucesso empresarial.

A área do planeamento operacional, e especificamente o tema do de máquinas, tem recebido uma atenção considerável por parte da científica. A prová-lo está o lançamento recente (1998) de uma publicação

científica dedicada ao assunto, o *Journal of Scheduling* [17], e a página WWW mantida por Peter Brucker [5].

Esta área também tem sido objecto de revisões bibliográficas regulares. Para citar apenas alguns exemplos, refere-se Graham, Lawler, Lenstra & Rinnooy Kan (1979) [8], e as bibliografias anotadas de Lenstra & Rinnooy Kan (1985) [11] e de Hoogeveen, Lenstra & Van de Velde (1997) [9].
A tendência poderá ser a cada vez maior autonomia do agendamento e do sequenciamento em relação à optimização combinatória.

1.1 Objectivos

O objectivo subjacente à realização desta dissertação é o desenvolvimento de um algoritmo que efectue o agendamento automático da produção, com vista à sua inserção num sistema de apoio à decisão. O algoritmo a desenvolver não necessita de produzir soluções óptimas, até porque, como se demonstrará, o problema é de grande complexidade computacional. Deverá produzir soluções de boa qualidade num tempo aceitável.

Um grande desafio que foi colocado foi a abordagem ao problema. Como se partiu de um problema prático de uma empresa real, foi necessário algum esforço na sua delimitação. Do sistema que estava a ser analisado, era necessário distinguir o essencial do acessório, de modo a poder construir um algoritmo que, sem a ambição de resolver todos os pequenos problemas, pudesse dar respostas de boa qualidade às questões essenciais.

1.2 Contribuições

Do trabalho realizado, destaca-se a análise e derivação de uma estratégia de solução e formalização de um modelo para tratar problemas de máquinas paralelas com matrizes de compatibilidade entre produtos e processadores, utilizando uma ampliação do modelo de Horn [10] para o planeamento com

preempção, de modo a suportar as referidas matrizes. Não foi encontrado literatura nenhum modelo semelhante.

Destaca-se também a aplicação da heurística PDEDD e o desenvolvimento de uma heurística de trocas para melhorar a qualidade das soluções, bem como a avaliação da complexidade computacional destas.

Uma grande fatia do tempo de trabalho foi dedicada à programação dos algoritmos desenvolvidos recorrendo a uma linguagem de alto nível. A aplicação computacional assim obtida pôde ser utilizada para a realização de testes computacionais aos algoritmos desenvolvidos. Estes testes incluíram a avaliação das melhorias introduzidas pelas heurísticas nas soluções das instâncias testadas.

1.3 Estrutura da dissertação

O segundo capítulo deste trabalho é dedicado à descrição detalhada do sistema produtivo da empresa, de modo a proporcionar uma visão integrada do problema em análise. De facto, em vez de apresentar apenas a parte do processo produtivo que será objecto do estudo, é apresentado todo o processo produtivo, de modo a que o leitor possa perceber melhor de onde surgem algumas das restrições e para que possa situar a sua importância relativa.

Uma vez apresentado o problema, procede-se no terceiro capítulo a uma análise mais detalhada do problema, questão a questão. É também feito um esforço inicial no sentido da formalização das principais questões que são colocadas. O material exposto neste capítulo servirá de base ao desenvolvimento de metodologias de resolução que se faz no quinto e no sexto capítulo.

No quarto capítulo faz-se uma revisão da literatura existente sobre operacional e sobre agendamento da produção, que tenha relevância para o

problema em análise. Assim, são apresentadas as principais metodologias estarão na base das soluções propostas mais tarde, em capítulos posteriores.

O quinto e o sexto capítulo constituem o núcleo do presente trabalho. No quinto capítulo o modelo é levado ao máximo grau de detalhe e é formalmente descrito através de um conjunto de expressões matemáticas. São propostos dois modelos de abordagem para o problema. Um modelo é baseado em programação matemática e comporta todas as restrições do problema, tornando-o excessivamente complexo. No outro modelo são relaxadas algumas restrições de modo a obter um modelo que possa ser resolvido.

O sexto capítulo vem complementar o capítulo anterior. Nele são desenvolvidas duas heurísticas capazes de tomar em consideração algumas restrições que anteriormente tinham sido relaxadas, contribuindo para a melhoria das soluções produzidas pelo modelo.

No sétimo capítulo é descrita a implementação computacional desenvolvida. Neste capítulo são explicadas as principais características da implementação, e são mostradas as principais funcionalidades da mesma. Nesta aplicação computacional são implementados os algoritmos apresentados no capítulo anterior.

O oitavo capítulo é dedicado à apresentação dos testes computacionais e à análise dos respectivos resultados. Os testes foram realizados recorrendo à implementação computacional desenvolvida e pretendem dar uma ideia da qualidade dos algoritmos desenvolvidos.

Finalmente, no nono e último capítulo são feitas algumas considerações são sintetizadas as principais conclusões do trabalho realizado. São perspectivados possíveis desenvolvimentos do presente trabalho, que lhe

conferir o verdadeiro interesse prático para a empresa onde o trabalho foi baseado.

Capítulo 2
DESCRIÇÃO GERAL DO SISTEMA

Neste capítulo será feita uma descrição detalhada do sistema produtivo utilizado pela empresa onde o trabalho foi desenvolvido. O objectivo desta descrição é proporcionar uma visão integrada dos processos da empresa. Começar-se-á por fazer uma descrição da empresa, seguida pela descrição do produto e, por fim, a descrição do processo produtivo.

2.1 A empresa

Trata-se de uma PME do sector têxtil que se dedica ao fabrico e comercialização de peúgas de homem. A empresa opera no mercado nacional, mas a maior fatia da produção é destinada à exportação.

A empresa faz grandes investimentos na evolução tecnológica. A título de exemplo, toda a movimentação de materiais é feita automaticamente e todo o processo é continuamente monitorizado. A cada momento é possível saber em detalhe a situação de cada encomenda (quantidade já fabricada), o rendimento de cada trabalhador ou a situação de cada máquina. Isto é conseguido através de unidades de aquisição de dados colocadas na maquinaria e posterior centralização da informação.

Os produtos da empresa podem ser classificados numa gama média-alta e a empresa aposta em dois factores fundamentais: a constante actualização em termos de colecções e os prazos de fornecimento curtos.

Com um determinado cliente (uma cadeia de lojas de retalho) a empresa comprometeu-se em prazos de entrega de 24 horas. As encomendas são recebidas por EDI (*electronic data interchange*) em determinado dia da semana, devendo ser expedidas no dia seguinte. Para já, este sistema apenas é com um cliente e apenas para fazer a reposição de stocks de encomendas de maior dimensão efectuadas anteriormente. No entanto, os volumes de cada reposição ainda são significativos.

Existe um grande controlo nos custos indirectos. Por exemplo, a empresa apenas possui dois funcionários administrativos. Todos estes factores têm permitido à empresa manter-se num mercado muito competitivo, onde tem que se confrontar com grandes concorrentes internacionais, principalmente de países asiáticos e de países pouco desenvolvidos.

2.2 O produto

Tal como já foi referido, a empresa fabrica peúgas de homem, que é o único produto fabricado pela empresa. Existem diferentes tipos de peúgas (lisas, com relevos, com desenhos, etc.) que exigem processos de confecção com algumas diferenças entre si, principalmente ao nível da tricotagem. Para além dos tipos de peúga há ainda a considerar três características importantes: os tamanhos, as cores e os materiais.

As peúgas são produzidas em três tamanhos. Dentro de cada encomenda, a distribuição pelos diversos tamanhos segue, aproximadamente, uma curva de Gauss. Conforme se irá ver, os diferentes tamanhos existentes são responsáveis por alguns tempos de preparação nas diversas máquinas.

Cada peúga é igualmente produzida em várias cores. Tal como os tamanhos, mudanças de cores também originam tempos de preparação em algumas máquinas.

O material em que a peúga é produzida também tem influência sobre o processo produtivo, havendo máquinas que reagem melhor que outras a determinados materiais. Este facto traduz-se numa melhor qualidade da peúga produzida, se esta for executada nas máquinas mais compatíveis com o material.

A título de exemplo, mostra-se na tabela seguinte a composição de uma ordem de fabrico típica. Para uma melhor compreensão, os códigos dos materiais e das cores foram substituídos por descrições.

Materiais			**Tamanhos**			**Total**
Seda	**Lycra**	**Mousse**	**10 ½**	**11**	**11 ½**	
Azul-1	Azul-2	Azul-3	900	1,650	750	**3,300**
Cinza-11	Cinza-12	Cinza-13	900	1,650	750	**3,300**
Cinza-21	Cinza-22	Cinza-23	300	450	300	**1,050**
Cinza-31	Cinza-32	Cinza-33	1,800	3,000	1,500	**6,300**
Castanho-1	Castanho-2	Castanho-3	300	450	300	**1,050**
		Total	**4,200**	**7,200**	**3,600**	**15,000**

Neste arquétipo de ordem de fabrico pode observar-se a especificação dos materiais a utilizar e as respectivas cores. Para cada combinação materiais/cores são especificadas as quantidades a produzir em cada tamanho. Uma ordem de fabrico concreta também especificaria as gamas operatórias e o tipo de máquinas a utilizar.

2.3 O processo

O processo produtivo das peúgas é constituído por cinco fases principais: tricotagem, remalhagem, enformagem, emparelhamento e embalagem.

Antes de poder ser iniciada a primeira fase do processo produtivo é proceder à aquisição e armazenagem dos materiais necessários à confecção

peúgas. Como as encomendas são estabelecidas com bastante antecedência, possível prever com rigor as necessidades, de modo a fazer uma boa gestão existências de matérias-primas.

O armazém é constituído por um conjunto de contentores colocados em alvéolos específicos. Cada contentor, por norma, contém determinado material numa determinada cor. Caso seja necessário guardar diferentes materiais no mesmo contentor, há o cuidado de escolher cores distintas. Todo o material é constituído por fio que se apresenta em bobines aptas para a utilização nos teares.

O material é movimentado por um AGV (*automated guided vehicle*). Este AGV faz a alimentação do parque de teares, levando os materiais necessários do armazém para os operadores dos teares e retornando o restante material ao respectivo alvéolo de armazenamento.

Para efeitos de controlo de existências, o AGV está equipado com um sistema de pesagem automático. Deste modo é possível saber, a cada momento, as quantidades de cada material existentes em armazém.

2.3.1 Tricotagem

A tricotagem da peúga é feita num tear automático. À saída do tear, a peúga é um tubo de tecido aberto em ambas as extremidades. O formato definitivo da peúga é obtido em fases posteriores do processo produtivo.

A secção de tricotagem é constituída por um parque de teares de diferentes tipos, embora funcionalmente semelhantes. Cada tear pode produzir um ou mais tipos de peúgas, o que origina uma matriz de compatibilidade entre produtos e teares.

Por razões relacionadas com a variabilidade do funcionamento das máquinas, dois teares tecnicamente iguais podem produzir o mesmo

com níveis de qualidade significativamente diferentes, aparecendo ocasionais na produção de algumas máquinas. Este facto deve ser tido consideração na altura de escolher o tear a utilizar em determinado isso, é mantido um cadastro para cada tear.

Partes de um mesmo lote podem ser tricotadas em teares de diferentes tipos, uma vez que não há diferenças nos produtos finais. Por outro lado, os tempos de produção de um mesmo produto em diferentes tipos de teares não são significativamente diferentes.

O parque de teares é assistido por operadores que estão encarregados de alimentar os teares com o material trazido pelo AGV do armazém e por efectuar todas as preparações necessárias nos teares.

Os tempos de preparação são dependentes da sequência de lotes a produzir em cada tear. Por exemplo, mudar de tamanho dentro do mesmo produto e da mesma cor é muito mais rápido que mudar de produto ou de fio (tipo ou cor).

Os operadores também são responsáveis por retirar as peúgas já tricotadas dos teares e pela sua colocação em contentores, de modo a poderem seguir para as fases posteriores do processo produtivo. Devido a este facto, é conveniente que o mesmo lote seja produzido em teares fisicamente próximos.

2.3.2 Remalhagem

Depois de tricotadas, as peúgas são enviadas em contentores para a secção de remalhagem. Mais uma vez, esta operação de transporte é feita automaticamente através de AGV's. Eventualmente, em casos mais ou menos excepcionais, pode haver aqui uma operação de armazenamento.

A remalhagem consiste em fechar uma das aberturas do tubo de tecido produzido no tear através da operação de cerzir. Esta operação é feita por um operador numa máquina de cerzir.

Cada estação de serviço (conjunto de um operador de uma máquina de cerzir) possui um *buffer* de entrada e um *buffer* de saída. O *buffer* de entrada é alimentado pelo AGV e é de onde o operador retira as peúgas a cerzir. O operador coloca as peúgas já cerzidas num contentor temporário que, uma vez cheio, é deslocado para o *buffer* de saída, de onde é removido pelo AGV, que o transporta para a operação seguinte. Os tempos de preparação das máquinas de cerzir são desprezáveis, pois envolvem apenas uma troca de fio.

2.3.3 Enformagem

Quando a peúga sai da secção de remalhagem, o seu aspecto é o de um tubo fechado numa das extremidades. Na secção de enformagem é conferido à peúga o aspecto físico de um pé. A peúga adquire a forma de um molde através de uma operação realizada com vapor.

A máquina de enformagem exige uma preparação quando se muda de tamanho. Por este facto, é conveniente que cada contentor que chega à enformagem contenha apenas um tamanho, embora possa conter duas ou mais cores. Este facto deverá ser tido em consideração no carregamento dos contentores na secção de tricotagem.

2.3.4 Emparelhamento

O emparelhamento é imediato à enformagem, e é feito automaticamente máquina apropriada. A necessidade desta operação advém do facto de peúgas produzidas com o mesmo tamanho teórico poderem chegar a com pequenas diferenças de tamanho entre si. Por razões comerciais, é conveniência que estas diferenças não se notem entre peúgas do mesmo

emparelhamento consiste em colocar, no mesmo par, duas peúgas o idênticas possível.

2.3.5 Embalagem

Antes da expedição as peúgas devem ser embaladas. A embalagem varia muito de cliente para cliente, pois é feita de acordo com as especificações daquele e do mercado onde opera. Por exemplo, pode haver necessidade de colocar vários tipos de rótulos ou de colocar papel de seda no interior da peúga. O tamanho das embalagens também é variável.

2.3.6 Planeamento da produção

Foram visíveis dois tipos de planeamento: o planeamento de capacidade e o agendamento fino das ordens de fabrico.

O planeamento de capacidade é feito tomando em consideração a capacidade de produção agregada do parque de teares, que parece ser o gargalo do sistema. As datas de entrega das encomendas são negociadas com base neste planeamento.

Este planeamento não leva em linha de conta as capacidades de cada tipo de teares nem as exigências das encomendas já aceites. Como o planeamento é feito para cerca de 80% da capacidade máxima teórica do parque de teares, o agendamento fino das encomendas acaba por resolver os problemas relacionados com eventuais desequilíbrios.

Enquanto que as encomendas são aceites com uma antecedência média quatro meses, o agendamento fino da produção é feito para um cerca de um mês. Existe uma pessoa responsável pelo agendamento um dos afinadores dos teares. O agendamento é feito de acordo com as existentes: datas de entrega das encomendas, datas de disponibilidade

materiais, incompatibilidades entre produtos e teares e restrições secções a jusante.

Um factor muito importante no agendamento fino, é a experiência da pessoa que o elabora. De facto, como se irá demonstrar, o problema do agendamento fino na secção de tricotagem é de natureza combinatória. O recurso a regras mentais dadas pela experiência é uma ajuda preciosa para a sua resolução, nos moldes do planeamento actual.

2.4 Objectivos para um sistema de planeamento automático

Como o parque de teares é o gargalo do sistema (as secções a jusante podem sofrer ajustes de capacidade de maneira a não haver desequilíbrios) o principal objectivo para um novo sistema de planeamento é o agendamento automático das ordens de produção nos diversos teares.

O plano de produção gerado deve poder ser editado por um supervisor e deve ser constantemente actualizado com a informação proveniente das unidades de aquisição de dados dos teares.

O sistema de planeamento deve informar os afinadores dos teares qual a afinação a fazer em determinado tear sempre que este terminar a execução de uma ordem.

Com um sistema de planeamento deste género, a responsabilidade de teares a ordens de encomenda deixaria de ser dos afinadores. Por outro houver um bom algoritmo de agendamento, as soluções deverão ser de qualidade e não deverão existir erros de agendamento.

Capítulo 3
ANÁLISE DO SISTEMA E FORMALIZAÇÃO MODELO

Uma vez feita a descrição do sistema produtivo utilizado é chegada a altura de definir, em concreto, o problema que irá ser trabalhado, de fazer a sua descrição detalhada e fazer alguma formalização das questões a tratar. Neste capítulo serão tratados estes três aspectos.

3.1 Introdução

Tal como a maioria dos sistemas produtivos, o sistema descrito no capítulo anterior é um sistema complexo. Essa complexidade traduz-se na existência de secções interdependentes, onde as decisões tomadas em determinado ponto do processo produtivo têm reflexos em todo o sistema.

Num sistema com esta dimensão e complexidade, a modelização da totalidade do sistema seria uma tarefa árdua. Os resultados seriam incertos e, provavelmente, não proporcionais ao esforço envolvido.

Por outro lado, a experiência dos nossos interlocutores na empresa e a observação atenta do sistema levou rapidamente à descoberta do cerne do problema que é, sem qualquer dúvida, o agendamento da produção na secção de tricotagem.

De facto, o sistema está dependente da capacidade produtiva desta secção. Por vários motivos:

– A capacidade nominal da secção de tricotagem só é variável a longo prazo, pois é necessário um grande investimento em maquinaria e o espaço disponível nas actuais instalações é diminuto;

– Pelo contrário, todas as outras secções a jusante podem ser ajustadas à capacidade nominal da tricotagem, pois exigem investimentos menores e facilmente concretizáveis;

– Uma vez feito o planeamento da secção de tricotagem, o planeamento das secções a jusante é bastante simplificado;

– Uma vez que os tempos de preparação nas secções a jusante da tricotagem são insignificantes, a sequência de tarefas agendadas na tricotagem não irá condicionar a produtividade daquelas;

– Conforme se irá ver, algumas restrições importantes impostas pelas outras secções à tricotagem podem ser incluídas no modelo.

Por estes motivos, o modelo a desenvolver contemplará apenas o agendamento na secção de tricotagem, embora possa incluir restrições impostas exteriormente.

3.2 Descrição detalhada do problema

Nas subsecções seguintes será feita uma descrição detalhada de todos os aspectos do problema do agendamento detalhado da secção de tricotagem.

3.2.1 Matriz de adequação produto/máquina

Na secção de tricotagem existe um conjunto de teares, funcionalmente semelhantes, mas possuindo cada um características específicas que os tornam mais aptos para determinado tipo de peúga e menos aptos, ou até incompatíveis, para outros tipos de peúga.

O exposto no parágrafo anterior impõe desde logo uma matriz de compatibilidade entre produtos e máquinas. Este matriz deverá determinar se uma determinada referência (produto) pode ser produzida num determinado tear (máquina).

Devido à variabilidade típica das máquinas, já referida, em dois teares perfeitamente iguais, ao ser produzida uma mesma referência, pode resultar em níveis de qualidade diferentes. Por este motivo, as tarefas devem ser agendadas em teares que as possam produzir com alto nível de qualidade.

A já referida matriz de compatibilidade entre produtos e máquinas pode transformada numa matriz de adequação entre produtos e máquinas. A

produto/máquina irá corresponder um nível de adequação, que poderá variar de "inadequado" até "perfeitamente adequado", passando por um número arbitrário de níveis intermédios, que deverão ser definidos pelos técnicos da empresa.
Ao nível "inadequado" corresponderia uma situação de total devendo o agendamento ser impossível. Pelo contrário, ao nível adequado" corresponderia uma situação de adequação máxima, sendo o agendamento preferencial.

3.2.2 Tempos de preparação das máquinas

Sempre que, em determinada máquina, haja mudança de tarefa, aquela tem que ser sujeita a um processo de afinação. Esta afinação consiste em uma ou em várias tarefas, conforme o tipo de mudança a efectuar. Por exemplo, uma mudança de tamanho (mesma referência, mesma cor) requer uma pequena mudança no programa de controle numérico do tear. Outro exemplo, uma mudança de cor pode implicar a substituição de uma, duas ou três bobines de fio e a produção de 1 ou 2 pares de peúgas para verificar se o produto está a sair com qualidade ou é necessária alguma afinação extra. Ainda a título de exemplo, uma mudança de referência pode implicar uma combinação das operações descritas nos dois exemplos anteriores.

Os tempos de preparação descritos no parágrafo anterior podem ser colocados numa matriz produto/produto, representando cada valor o tempo de preparação de um tear quando se passa do produto 'x' para o produto 'y'.

A observação do sistema permitiu concluir que os tempos de preparação das máquinas são irrelevantes quando comparados com os tempos de produção efectivos de uma encomenda. Isto poderá significar que a optimização dos tempos de preparação é uma questão secundária.

3.2.3 Datas de disponibilidade e de conclusão

Cada uma das tarefas a executar impõe à secção de tricotagem uma data de conclusão que deve ser cumprida. Por norma, as tarefas não possuem datas de disponibilidade. No entanto, pode haver imposição de datas de disponibilidade por parte do armazém (inexistência de matérias primas necessárias). Uma data de disponibilidade deste tipo afectaria todas as encomendas que necessitassem da matéria prima em falta. É uma situação perfeitamente possível mas extraordinária.

3.2.4 Preempção

As tarefas são completamente preemptivas: podem ser interrompidas a qualquer altura e podem ser retomadas, no mesmo tear, ou em qualquer outro permitido pela matriz de adequação produto/máquina. O processamento paralelo também é admissível: quanto maior for o número de teares envolvidos na execução de uma encomenda, mais rapidamente esta será concluída.

3.2.5 Estado inicial

O problema também é caracterizado por um estado inicial do parque de teares: num determinado momento, todos os teares possuem uma afinação para um determinado produto e podem estar a executar determinada encomenda ou estar parados.

3.2.6 Horizonte temporal do agendamento

O horizonte temporal do agendamento deve ser decidido pelos técnicos da empresa e pode ir desde um período de tempo relativamente curto (1 semana, 1 mês) até ao agendamento provisório de todas as encomendas em carteira (4 a 6 meses).

Como o problema é, a nível computacional, bastante complexo, poderão surgir problemas com horizontes temporais muito grandes. Por outro lado, o mais importante no agendamento é o "agora" e quanto mais distantes no tempo estiverem as encomendas, menos influência terão sobre a solução de agendamento.

3.2.7 Velocidade de processamento

A velocidade dos teares é homogénea, ou seja, não existem diferenças significativas de tear para tear. No entanto, em teoria, pode admitir-se que haja diferenças de velocidade entre os teares. Por exemplo, um tear antigo pode produzir mais lentamente que um tear novo que utilize uma tecnologia mais recente.

A velocidade de confecção de uma peúga num tear também é homogénea de produto para produto. Mais uma vez, em teoria, é admissível velocidades diferentes na execução de dois produtos diferentes. Por exemplo, um produto pode necessitar de uma mistura especial de diferentes fios que seja de confecção mais demorada que o normal.

Segundo o mesmo raciocínio, é admissível, do ponto de vista teórico, que a velocidade de execução possa ser resultado de uma interacção entre o produto e o tear utilizado na sua confecção. Tal interacção seria de difícil quantificação, pois necessitaria de ferramentas estatísticas poderosas, de experiências desenhadas para a sua identificação e, ao contrário dos pressupostos anteriores, a sua introdução num modelo de resolução seria muito complexa. Por estes motivos, não será considerada.

3.2.8 Sortidos

Por vezes há a necessidade de produzir o que na empresa se denomina "sortido". Um "sortido" é uma forma especial de embalagem em que mais referências (ou cores) diferentes devem ser anexadas. Por exemplo, fornecedor especifica que cada embalagem deve conter dois pares de específicas diferentes. Isto implica que, no momento da embalagem, presentes os vários componentes do sortido. Se não se quiser recorrer armazenamento intermédio, é de toda a conveniência que os diversos componentes do sortido sejam produzidos de forma mais ou menos

3.2.9 Localização física das máquinas utilizadas

De modo a facilitar o trabalho dos operadores que assistem os teares e dos AGV, seria conveniente que os teares utilizados na produção de determinada ordem de fabrico fossem fisicamente próximos.

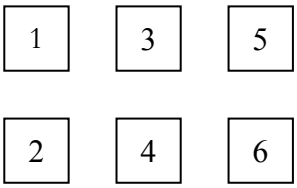
3.3 Formalização do problema

Feita a descrição do problema, é chegada a altura de fazer alguma formalização das questões que vai ser necessário analisar. A exposição será acompanhada de um exemplo de pequenas dimensões, que servirá para ilustrar os vários aspectos do problema.

Nesta secção não será introduzida qualquer notação, ficando essa questão para a formulação matemática que irá ser feita no capítulo 5.

3.3.1 Máquinas e produtos

Considere-se o seguinte exemplo: a secção de tricotagem da empresa possui um parque de máquinas composto por 6 teares idênticos e que estão implantados no terreno tal como é indicado na figura.



Os teares operam a uma cadência de 60 pares de peúgas por hora, excepto o tear 1, que opera a 54 pares por hora, e o tear 6, que opera a 66 pares por hora. Se se considerar que os teares 1 e 6 são excepções ao normal pode dizer-se que o tear 1 funciona a 90% do normal e o tear 6 funciona a 110% do normal. Neste caso, o "normal" é definido como a velocidade mais frequente, mas pode ser definido arbitrariamente.

Admita-se agora que o catálogo da empresa é constituído por 4 produtos, que são fabricados em 4 cores e em 3 tamanhos, num total de 48 combinações possíveis. Considere-se que o tempo de processamento de um par de peúgas do tamanho intermédio é considerado "normal" e que, num tear a funcionar a velocidade normal, é de 1 minuto. O tamanho mais pequeno demora 95% do tempo normal (57 segundos) e, o tamanho maior, demora 105% do tempo normal (63 segundos). Mais uma vez, a definição de "normal" é arbitrária.
Para sistematizar melhor a questão da velocidade de processamento, admita-se que se define uma constante como o "tempo de processamento de uma unidade de produto normal num tear a funcionar à velocidade normal". No caso concreto, seja 60 segundos o valor dessa constante. Por exemplo, o tempo unitário de processamento de um produto que demore 95% do tempo normal, num tear a funcionar a 90% da velocidade normal, pode ser calculado da seguinte forma:

$$60 \cdot \frac{0.95}{0.90} = 63.[3] \text{segundos}$$

Se se admitir, como é o caso, que não há interacção entre o produto e a máquina utilizada, que possa modificar a velocidade de processamento, a expressão anterior é válida para qualquer par produto/máquina. A duração de uma tarefa pode então ser expressa em número de unidades a produzir.

Entre as tarefas e as máquinas podem existir incompatibilidades ou inadequações. Admita-se que a empresa definiu quatro níveis de adequação produto/máquina, que são os seguintes:

0 - Totalmente incompatível (agendamento impossível);

1 - Agendar em último caso;

2 - Agendar como segunda opção;

3 - Perfeitamente compatível (agendamento preferencial);

Foi também construída uma matriz com as adequações para os vários pares máquina/produto, que se apresenta a seguir.

	Máquina					
Produto	**1**	**2**	**3**	**4**	**5**	**6**
1	0	0	2	2	3	3
2	1	1	0	0	3	3
3	3	3	3	3	0	0
4	0	0	0	0	3	3

A interpretação da matriz é trivial. Por exemplo, o produto 1 não pode ser agendado nas máquinas 1 e 2. Pelo contrário, o produto 4 deve ser agendado nas máquinas 5 e 6, uma vez que não é permitido agendar em mais nenhuma. O produto 2 deve ser agendado nas máquinas 5 e 6, embora, em último caso, possa ser agendado nas máquinas 1 e 2.

Outra questão que se coloca são os tempos de preparação das máquinas. Estes tempos são mais ou menos uniformes, independentemente das máquinas e dos produtos envolvidos. Admita-se que, para trocar de produto ou de cor, com ou sem mudança de tamanho, se incorre num tempo de preparação de 15 minutos. Para apenas trocar de tamanho são necessários 5 minutos.

3.3.2 Estado inicial

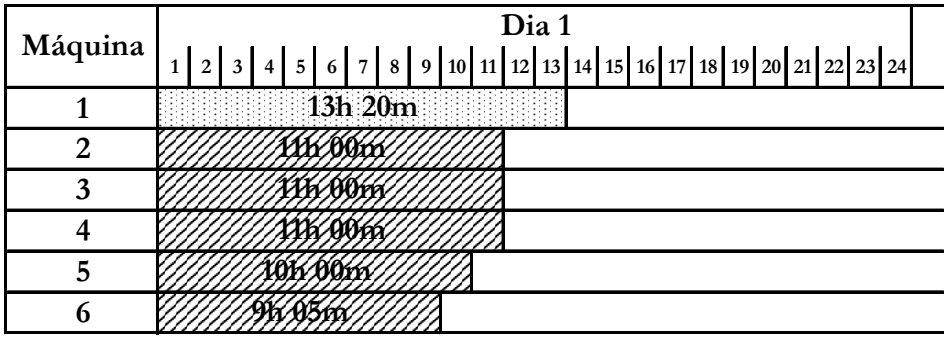
Como já foi dito, o sistema é caracterizado por um estado inicial. Esse estado pode ser expresso em termos das tarefas que cada máquina se encontra a realizar e do tempo (ou quantidade) em falta para a conclusão da ordem de fabrico em curso. Igualmente, a máquina pode não estar a executar nenhuma tarefa. Pode estar em manutenção ou, simplesmente, descarregada. Por outro lado, caso uma máquina esteja ocupada, deverá ser possível especificar se a tarefa em curso pode ser interrompida ou se deve ser considerada definitivamente agendada, sendo a capacidade em causa subtraída à máquina.

Essas condições podem ser resumidas numa tabela semelhante à que se onde foram colocados dados relativos ao exemplo que vem a ser

Máquina	Tarefa em curso	Quantidade em falta	Interrupção
1	OE 1	720	Sim
2	OE 2	660	Não
3	OE 2	660	Não
4	OE 2	660	Não
5	OE 3	600	Não
6	OE 3	600	Não

Por exemplo, a máquina 1 está a executar a ordem de encomenda 1 (OE 1), na qual falta executar 720 unidades. A tabela especifica ainda que a tarefa em curso na máquina 1 pode ser agendada de forma diversa. Pelo contrário, todas as outras tarefas devem ser consideradas definitivamente agendadas.

Estas restrições podem ser colocadas sob a forma de diagrama de Gantt, para uma melhor visualização.



Os sombreados representam as tarefas em curso nas diversas máquinas. Como a tarefa da máquina 1 pode ser agendada novamente, o sombreado é diferente.

Os valores inscritos em cada um dos sombreados referem-se aos tempos exactos que as máquinas demorarão a completar as tarefas e são calculados pela expressão deduzida na secção anterior. Por exemplo, para a máquina 1 temos:

$$(720\ pares)\cdot(1\ min)\cdot\frac{1.00}{0.90}=800\ min=13h\ 20m$$

Recorde-se que a máquina 1 funciona a 90% da velocidade normal, e admita-se que se trata do tamanho intermédio (tamanho 2), cujo tempo de produção é 100% do tempo normal.

3.3.3 Ordens de encomenda

Uma ordem de encomenda deve especificar, no que se refere ao agendamento, o produto, a cor, o tamanho, a quantidade e a data de conclusão a respeitar.

No agendamento, as ordens de encomenda que estão em curso, e cuja interrupção e novo agendamento é permitido, podem ser consideradas como ordens de encomenda normais, sendo o tempo de preparação da máquina igual a zero, caso voltem a ser agendadas da mesma forma.

Assim, poderia ter-se uma situação como a da tabela seguinte.

Ordem	Produto	Cor	Tamanho	Qtd.	Conclusão
OE 1	3	1	2	720	1
OE 2	3	4	2	-	-
OE 3	4	2	2	-	-
OE 4	1	1	2	3,000	4
OE 5	1	2	2	6,000	4
OE 6	2	3	2	3,000	4
OE 7	2	4	2	9,000	5
OE 8	4	2	2	6,000	6
OE 9	4	3	2	9,000	7

Considere-se que o tamanho 2 é o tamanho intermédio, cuja produção demora o tempo normal.

Repare-se na ausência de quantidades nas ordens de encomenda 2 e 3. estas ordens foram consideradas definitivamente agendadas, a falta já foi especificada no estado inicial do sistema. Apenas aparecem nesta tabela para que se possam calcular tempos de preparação para as máquinas envolvidas.

Admita-se que se está a fazer o planeamento para uma semana (7 dias) e que as datas de conclusão se referem ao dia em que a encomenda deve estar concluída. Por exemplo, a ordem de encomenda 7 deve ser concluída até ao final do dia 5.
Pode agora construir-se um conjunto de diagramas de Gantt de forma a reflectir as novas restrições. Estes diagramas estão elaborados para cada uma das máquinas e assinalam com sombreado os períodos onde não é permitido agendar cada uma das ordens de encomenda.

Máquina 1

Ordem	dia 1	dia 2	dia 3	dia 4	dia 5	dia 6	dia 7
OE 1							
OE 4							
OE 5							
OE 6							
OE 7							
OE 8							
OE 9							

Máquina 2

Ordem	dia 1	dia 2	dia 3	dia 4	dia 5	dia 6	dia 7
OE 1							
OE 4							
OE 5							
OE 6							
OE 7							
OE 8							
OE 9							

Poderia ser elaborado um diagrama deste tipo para todas as máquinas. As ordens de encomenda 2 e 3 não são listadas porque já estão agendadas de forma definitiva.

Repare-se que, como as máquinas 1 e 2 têm o mesmo conjunto de adequações produto/máquina, o diagrama é semelhante, com excepção do período inicial, em que a máquina 2 está ocupada com a ordem de fabrico 2.

3.3.4 Outras restrições

Para além de tudo o que já foi considerado o modelo poderia ainda incluir outras restrições.
Considere-se que o fio de cor 3 está esgotado em armazém e que só chega a meio do dia 2. As encomendas 6 e 9, que utilizam esse fio, só se podem iniciar nesse momento. Esta restrição dá origem a datas de disponibilidade diferentes do momento inicial do agendamento.

Mostra-se agora o diagrama de Gantt correspondente à máquina 1, em que esta última restrição já é considerada.

Máquina 1

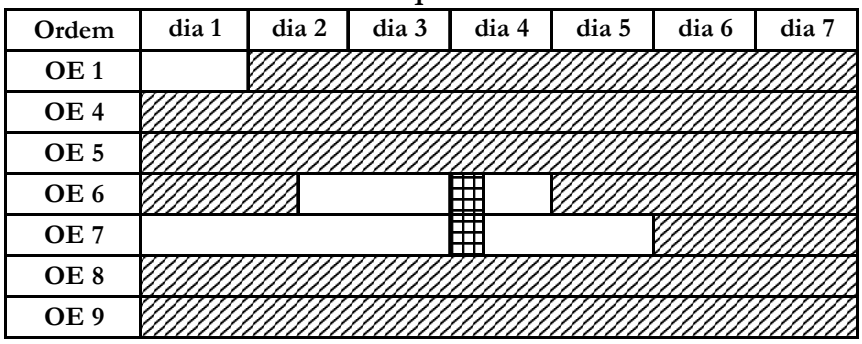
Ordem	dia 1	dia 2	dia 3	dia 4	dia 5	dia 6	dia 7
OE 1							
OE 4							
OE 5							
OE 6							
OE 7							
OE 8							
OE 9							

A encomenda 6 já só pode ser agendada a partir do meio do segundo dia. No caso concreto da máquina 1, a encomenda 9 já não podia ser agendada por incompatibilidade produto/máquina.

Considere-se também que no dia 4 vai haver manutenção programada equipamentos. Assim, as máquinas 1 e 2 terão que estar paradas nas horas do dia, as máquinas 3 e 4 pararão nas 8 horas seguintes e as nas seguintes. Durante estes períodos não deve ser agendada qualquer encomenda para as máquinas afectadas.

Mais uma vez se mostra o diagrama de Gantt para a máquina 1, em que esta última restrição já foi reflectida.

Máquina 1



Poder-se-ia ainda adicionar os sortidos ao conjunto de restrições. Considere-se que as ordens 4 e 6 fazem parte de um sortido, devendo ser fabricadas em paralelo. Ao contrário das restrições anteriores, esta restrições não pode ser imposta por limitação dos períodos de agendamento. A solução passa pela criação de uma penalidade para os planos de agendamento onde as duas ordens tenham períodos de fabrico diferentes.

3.3.5 Penalidades

Tal como já ficou implícito, num sistema deste tipo há restrições que não podem ser violadas sob qualquer circunstância, e há restrições que podem ser violadas incorrendo em algum tipo de penalidade.

No primeiro tipo incluem-se as restrições que garantem a integridade do tais como, a que garante que numa mesma máquina e num mesmo apenas possa estar agendada uma tarefa. Por exemplo, uma data de disponibilidade imposta por uma falta de matérias-primas também não poder ser violada.

Uma data de conclusão já pode fazer parte do segundo conjunto de restrições, na medida em que pode haver necessidade de a violar, embora incorrendo na referida penalidade. Por exemplo, se houve uma má negociação da data de conclusão e esta não puder ser cumprida por falta de capacidade, deve haver a possibilidade de a violar.
Dentro deste conjunto de restrições, há as mais importantes, que penalidades significativas, e há as menos importantes, que acarretam

inconvenientes para o sistema. As datas de conclusão, por norma, implicam penalidade elevadas, que podem ser contratuais ou de desgaste de imagem da empresa. Por outro lado, se o fabrico dos sortidos não for feito em paralelo, haverá o inconveniente da armazenagem intermédia e a importância da restrição dilui-se se houver espaço de armazenagem em abundância.
O conjunto de restrições que podem ser violadas deve ser ordenado pela sua importância relativa, pelos técnicos da empresa. Esta ordenação dará origem a um sistema de penalidades, a introduzir no algoritmo de agendamento, que reflicta a importância atribuída a cada tipo de restrição.

3.3.6 Conclusão

Conforme ficou mais ou menos explícito nas páginas anteriores, o problema apresenta um conjunto variado de restrições que é necessário considerar no desenvolvimento de um método de resolução.

No entanto, há 4 restrições que se revelam mais importantes, e que, por si só, implicam a existência ou não de uma solução admissível:

1 - A capacidade das máquinas – fundamentalmente, é limitada pela velocidade da máquina, ou seja, para um mesmo horizonte de planeamento, as máquinas mais rápidas têm mais capacidade. As de tempo (por exemplo, para manutenção) também reduzem a capacidade disponível.

2 - A dimensão da encomenda – pode considerar-se que o facto de haver produtos de fabrico mais demorado aumenta a dimensão de uma encomenda, no sentido em que aumenta a capacidade necessária para a elaborar;

3 - As incompatibilidades produto/máquina – são uma restrição forte, pois diminuem consideravelmente o espaço de soluções de agendamento;
4 - As datas de disponibilidade e de conclusão – também reduzem consideravelmente o espaço de soluções do problema, pois impõem que

só uma parte da capacidade disponível possa ser utilizada por cada encomenda.

A esta lista poder-se-iam juntar os tempos de preparação das máquinas, como consumidores de capacidade. Apesar de, efectivamente, serem consumidores de capacidade disponível, os valores utilizados são muito pequenos, quando comparados com o tamanho das ordens de encomenda. No entanto, um método de resolução deve procurar a minimização destes tempos de preparação.

Todas as outras restrições não são redutoras do espaço de soluções: são indicativas da qualidade de uma solução.

Capítulo 4
REVISÃO DE LITERATURA

Este capítulo consiste numa revisão aos textos que foram consultados, na área do planeamento operacional. Adicionalmente, será introduzida a notação mais frequentemente utilizada nos textos sobre o assunto (notação de Graham).

4.1 Os problemas de planeamento operacional

Os problemas de planeamento operacional dizem respeito, fundamentalmente, ao agendamento das tarefas produtivas (ordens) em um ou vários processadores (máquinas), respeitando diversas restrições. O texto desta secção segue de perto Blazewicz et al. [3].

4.1.1 Caracterização genérica

Genericamente, cada problema pode ser identificado por três conjuntos principais: o conjunto $T = \{T_1, T_2, \cdots, T_n\}$, constituído pelas n tarefas a conjunto $P = \{P_1, P_2, \cdots, P_m\}$, constituído pelos m processadores, e o $R = \{R_1, R_2, \cdots, R_s\}$, constituído por s tipos de recursos adicionais. O

planeamento operacional consiste em atribuir processadores do conjunto P e, eventualmente, recursos do conjunto R às tarefas do conjunto T, de modo a que estas sejam completadas respeitando todas as restrições impostas.
Há duas restrições genéricas que toda a teoria clássica sobre operacional impõe aos sistemas: a primeira especifica que um determinado instante, apenas pode estar a processar uma única tarefa; a impõe que uma tarefa, em determinado instante, apenas pode ser um único processador, podendo esta última restrição ser relaxada em casos concretos, como é o caso do exemplo em estudo, já apresentado no capítulo anterior.

4.1.2 Caracterização dos processadores

Os processadores podem ser classificados em "paralelos", quando executam as mesmas funções, ou "dedicados", quando são especializados na execução de determinadas tarefas.

Dos processadores paralelos distinguem-se três tipos, quanto à velocidade de processamento. Se as velocidades dos vários processadores forem iguais estamos perante processadores "idênticos". Se as velocidades forem diferentes, cada processador com uma velocidade b_i, mas constante e independente da tarefa executada, designam-se por processadores "uniformes". Finalmente, se a velocidade do processador depender da tarefa a realizar, são os chamados processadores "não relacionados".

No caso de processadores dedicados, há três modos de processamento o *open shop*, o *flow shop* e o *job shop*. No modo *open shop*, cada tarefa têm processada em todas as máquinas, sendo a ordem arbitrária. O modo idêntico, mas há uma ordem de processamento estabelecida. No modo cada tarefa deve ser processada num número arbitrário de máquinas. sistemas, pode ou não ser autorizada uma espera entre dois consecutivos, dependendo do tamanho dos espaços de armazenamento

processadores. Se este espaço não existir, não pode haver espera entre processamentos consecutivos.

4.1.3 Caracterização das tarefas

Cada tarefa $T_j \in T$ é caracterizada pelo seguinte conjunto de informações:

1 - Vector de tempos de processamento, $p_j = \left[p_{1j}, p_{2j}, \cdots, p_{mj}\right]^T$, em que p_{ij} representa o tempo que o processador P_i demora a processar a tarefa T_j. Se os processadores forem idênticos, fica $p_{ij} = p_j$ para $i = 1,2,\cdots,m$. No caso de processadores uniformes fica $p_{ij} = p_j / b_i$, com $i = 1,2,\cdots,m$, sendo p_j um tempo de processamento padrão (normalmente medido no processador mais lento) e sendo b_i o factor de velocidade de processamento para o processador P_i.
2 - Data de disponibilidade, r_j, que representa o momento a partir do qual a tarefa T_j pode ser processada. Um caso importante acontece quando $r_j = 0$ para todo o j.

3 - Data de conclusão, d_j, que especifica o momento em que a tarefa T_j deverá estar concluída.

4 - *Deadline*, $\tilde{d}_j$, que impõe uma data de conclusão obrigatória, sem possibilidade de violação, para a tarefa T_j.

5 - Peso da tarefa, w_j, que reflecte a prioridade, importância, ou urgência da tarefa T_j.

6 - Eventuais recursos necessários.

Geralmente, é assumido que todos os parâmetros p_j, r_j, d_j, $\tilde{d}_j$ e w_j são valores inteiros, ou, pelo menos racionais.

Se as tarefas, uma vez iniciadas, puderem ser interrompidas e retomadas tarde, no mesmo ou em outro processador, está-se numa situação "com

preempção". Caso as tarefas, uma vez iniciadas, não possam ser interrompidas, está-se numa situação "sem preempção".
Se no conjunto de tarefas existirem condições do tipo $T_i \prec T_j$ que imponham que a tarefa T_i tem que estar completada antes que a tarefa T_j se possa iniciar, está-se perante um conjunto de tarefas "dependentes". Caso aquelas condições não existam, as tarefas denominam-se de "independentes". Normalmente, as relações de dependência são explicitadas num grafo apropriado.

4.1.4 Critérios de óptimo

Uma solução de planeamento operacional (um agendamento) consiste na afectação de processadores do conjunto P (e eventuais recursos do conjunto R) às tarefas do conjunto T de forma a que as seguintes condições sejam satisfeitas:

– Em cada momento, nenhum processador está afectado a mais do que tarefa e nenhuma tarefa decorre em mais do que um processador[1];

– A tarefa T_j é processada no intervalo $[r_j, \infty[$;

– Todas as tarefas são completadas;

– Todas as relações de precedência (se existirem) são respeitadas;

– Se o agendamento não permitir preempção, nenhuma tarefa é interrompida e se a preempção for permitida, o número de preempções finito[2];

– As restrições respeitantes a recursos são respeitadas.

A partir de um agendamento podem calcular-se com facilidade as seguintes quantidades:

– Instante de conclusão da tarefa, C_j;

[1] Tal como já foi referido, esta última condição pode ser relaxada.
[2] Apenas por motivos de ordem prática.

– Tempo de permanência no sistema (*flow time*), $F_j = C_j - r_j$, que é a soma dos tempos de espera e dos tempos de processamento;

– *Lateness*, $L_j = C_j - d_j$;

– Atraso (*tardiness*), $D_j = max\{L_j, 0\}$

A partir destas quatro quantidades podem ser desenvolvidas as principais medidas de eficiência ou critérios de óptimo:

– Tamanho do agendamento (*makespan*), $C_{max} = max\{C_j\}$;

– Tempo médio de permanência no sistema (*mean flow time*), $\overline{F} = \frac{1}{n} \cdot \sum_{j=1}^{n} F_j$, ou o tempo médio de permanência ponderado, $\overline{F}_w = \sum_{j=1}^{n} (w_j \cdot F_j) \Big/ \sum_{j=1}^{n} w_j$;

– *Lateness* máxima, $L_{max} = max\{L_j\}$;

– Atraso médio, $\overline{D} = \frac{1}{n} \cdot \sum_{j=1}^{n} D_j$, ou o atraso médio ponderado, $\overline{D}_w = \sum_{j=1}^{n} (w_j \cdot D_j) \Big/ \sum_{j=1}^{n} w_j$;

– Número de tarefas atrasadas, $U = \sum_{j=1}^{n} U_j$, em que $U_j = 1$ se $D_j > 0$ e $U_j = 0$ se $D_j \leq 0$, ou o número ponderado de tarefas atrasadas, $U_w = \sum_{j=1}^{n} w_j \cdot U_j$.

Entre outras características, o critério do tamanho do agendamento, conduz a altas taxas de ocupação do processadores e a que o conjunto processadores seja libertado no mais curto espaço de tempo. O critério médio de permanência no sistema é importante se se pretender a

dos trabalhos em curso. Os critérios relacionados com datas de particular importância quando se produz por encomenda, uma vez que, possibilidades de agendamento sem atrasos, a minimização destes encontrará esses agendamentos.

As medidas de eficiência não se esgotam nestas que foram apresentadas. A situação concreta pode aconselhar outras medidas que melhor de adeqúem aos objectivos do planeamento operacional.

4.1.5 Algumas definições

Um problema de planeamento operacional, Π, pode ser caracterizado pelo conjunto de parâmetros descritos nas secções anteriores e por uma medida de eficiência. Uma instância I do problema Π pode ser obtida atribuindo valores aos parâmetros do problema.

Um agendamento cujo valor da medida de eficiência, γ é mínimo é chamado "agendamento óptimo" e γ será denotada por γ^*.

Finalmente, um "algoritmo de agendamento" constrói uma solução para o problema de agendamento Π. O ideal seria encontrar algoritmos de optimização. No entanto, devido à complexidade computacional de muitos problemas, algoritmos de aproximação e heurísticas são perfeitamente aceitáveis.

4.1.6 A notação de Graham

Uma forma de catalogar os problemas de agendamento foi proposta por Graham em 1979 [8] e posteriormente complementada por outros autores. A notação é constituída por 3 campos referenciados como $\alpha|\beta|\gamma$.

O primeiro campo, $\alpha = \alpha_1\alpha_2$ descreve os processadores. O parâmetro $\alpha_1 \in \{\varnothing, P, Q, R, O, F, J\}$ caracteriza o tipo de processadores e tem o seguinte significado:

$\alpha_1 = 1$: um único processador;

$\alpha_1 = P$: processadores idênticos;

$\alpha_1 = Q$: processadores uniformes;

$\alpha_1 = R$: processadores não relacionados;

$\alpha_1 = O$: processadores dedicados, modo *open shop*;

$\alpha_1 = F$: processadores dedicados, modo *flow shop*;

$\alpha_1 = J$: processadores dedicados, modo *job shop*.

O parâmetro $\alpha_2 \in \{\varnothing, k\}$ representa o número de processadores no sistema:

$\alpha_2 = \varnothing$: número variável de processadores[3];

$\alpha_2 = k$: o número de processadores é k (sendo k um inteiro positivo).

O segundo campo, $\beta = \beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8$ descreve as características das tarefas e dos recursos. O parâmetro $\beta_1 \in \{\varnothing, pmtn\}$ indica a possibilidade de preempção:

$\beta_1 = \varnothing$: a preempção não é permitida;

$\beta_1 = pmtn$: a preempção é permitida.

O parâmetro $\beta_2 \in \{\varnothing, res\}$ caracteriza os recursos adicionais:

$\beta_2 = \varnothing$: não existem recursos adicionais;

$\beta_2 = res$: existem restrições relacionadas com recursos.

[3] O símbolo $\varnothing$ significa "vazio" e é omitido na apresentação dos problemas.

O parâmetro $\beta_3 \in \{\varnothing, prec, uan, tree, chains\}$ caracteriza as restrições de precedência:

$\beta_3 = \varnothing$: não existem relações de precedência – tarefas independentes;

$\beta_3 = prec$: restrições de precedência genéricas;

$\beta_3 = uan$: restrições de precedência do tipo *uniconnected activity networks*;

$\beta_3 = tree$: restrições de precedência em forma de árvore;

$\beta_3 = chains$: restrições de precedência que formam um conjunto de cadeias.

O parâmetro $\beta_4 \in \{\varnothing, r_j\}$ caracteriza as datas de disponibilidade das tarefas:

$\beta_4 = \varnothing$: todas as datas de disponibilidade são iguais a zero;

$\beta_4 = r_j$: as datas de disponibilidade variam com a tarefa.

O parâmetro $\beta_5 \in \{\varnothing, p_j = p, \underline{p} \le p_j \le \overline{p}\}$ caracteriza a duração das tarefas:

$\beta_5 = \varnothing$: as tarefas têm durações arbitrárias;

$\beta_5 = (p_j = p)$: as tarefas têm todas a duração p ;

$\beta_5 = (\underline{p} \le p_j \le \overline{p})$: a duração das tarefas está entre no intervalo $[\underline{p}, \overline{p}]$.

O parâmetro $\beta_6 \in \{\varnothing, \tilde{d}\}$ caracteriza as *deadlines*:

$\beta_6 = \varnothing$: não há *deadlines* no sistema, embora possa haver datas de conclusão;

$\beta_6 = \tilde{d}$: existem *deadlines* que devem ser obrigatoriamente respeitadas.

O parâmetro $\beta_7 \in \{\varnothing, n_j \le k\}$ caracteriza o número de processadores necessários para completar uma tarefa no caso do sistema *job shop*:

$\beta_7 = \varnothing$: o número de processadores necessários é arbitrário, ou o sistema não é *job shop*;

$\beta_7 = \left(n_j \leq k\right)$: o número de processadores necessários a cada tarefa não é superior a k.

Finalmente, o parâmetro $\beta_8 = \{\varnothing, no\text{–}wait\}$ descreve se são permitidas esperas ao programar processadores dedicados:

$\beta_8 = \varnothing$: o espaço de armazenamento entre processadores é ilimitado;

$\beta_8 = no\text{–}wait$: não há espaço de armazenamento entre processadores, pelo que, sempre que uma tarefa terminar o processamento num processador, deve ser imediatamente transferida para o processador que lhe sucede e o processamento deve ser aí iniciado.

O campo γ refere-se ao critério de eficiência a utilizar e pode ser escolhido de entre os que já foram referidos.

Por exemplo, o problema $P \| C_{max}$ refere-se ao agendamento sem tarefas independentes, com tempos de processamento arbitrários, que sistema no momento 0, em processadores paralelos idênticos, de modo minimizar o tamanho do agendamento. Outro exemplo, o problema $O3\left|pmtn, r_j\right|\sum C_j$ refere-se ao agendamento preemptivo de tarefas com de processamento arbitrários num *open shop* de 3 máquinas, com datas de disponibilidade arbitrárias para as várias tarefas, de forma a minimizar o médio de permanência no sistema[4].

[4] A minimização de $\sum C_j$ é equivalente à minimização $\overline{F}$,

Ao longo do presente texto, esta notação será a utilizada. Sempre que for necessário, para itens não contemplados nesta notação, introduzir-se-á no momento a notação necessária.

4.2 Os problemas de máquinas paralelas

Conforme se depreende do que foi referido na secção anterior, os problemas de máquinas paralelas são um caso particular dos problemas de planeamento operacional.

Uma vez que este tipo de problema ocorre frequentemente, é grande o interesse pelo seu estudo. Ainda em 1990 foi publicada por Cheng & Sin [6] uma revisão de literatura sobre o tema. Em Pinedo [16] são referidos vários casos para os quais existem resultados. Alguns desses casos, os que têm mais interesse para o presente trabalho são resumidos nos parágrafos que se seguem:

Para o problema $Pm||C_{max}$ existe a regra LPT, *longest processing time first*, em que, sempre que uma máquina é libertada, se agenda a tarefa com o maior tempo de processamento. A regra não garante a solução óptima, mas demonstra-se que a seguinte desigualdade é verdadeira:

$$\frac{C_{max}(LPT)}{C_{max}(OPT)} \leq \frac{4}{3} - \frac{1}{3m}$$

$C_{max}(LPT)$ é o valor da solução conseguida utilizando a regra LPT, e $C_{max}(OPT)$ é o valor da solução óptima. Como a desigualdade é verdadeira, é garantido que a regra LPT, sob nenhuma circunstância, produz uma solução com um valor superior em mais de 33% (aproximadamente) ao valor da solução óptima.

Uma generalização deste problema, com interesse relativo para o presente trabalho, surge quando cada tarefa apenas pode ser processada num

M_j do conjunto dos processadores, M. Considere-se o problema $Pm|p_j = 1, M_j|C_{max}$ em que os subconjuntos M_j estão encapsulados, ou uma, e só uma, das seguintes quatro condições é verdadeira para cada par de subconjuntos j e k.

– M_j é igual a M_k $\left(M_j = M_k\right)$

– M_j é subconjunto de M_k $\left(M_j \subset M_k\right)$

– M_k é subconjunto de M_j $\left(M_k \subset M_j\right)$

– M_j e M_k não se intersectam $\left(M_j \cap M_k = \varnothing\right)$

Nestas condições, há uma regra de despacho que produz soluções óptimas: a regra LFJ, *least flexible job first*. Segundo esta regra, sempre que uma máquina for libertada, deve ser agendada a tarefa menos flexível, isto é, aquela que pode ser processada no menor número de máquinas. Os empates são resolvidos arbitrariamente. Se os subconjuntos não forem encapsulados, a regra já não garante a solução óptima.

A regra LFJ não especifica que máquina deve ser utilizada quando duas ou mais máquinas são libertadas simultaneamente. Por este motivo, esta regra aparece muitas vezes associada à regra LFM, *least flexible machine first*. Como o nome indica, segundo a regra LFM, sempre que duas ou mais máquinas forem libertadas simultaneamente, a máquina menos flexível (aquela em que se pode agendar o menor número de tarefas ainda não agendadas) deve ser utilizada primeiro. É vulgar utilizar as regras LFJ e LFM em conjunto sob a designação de regra LFM-LFJ. As soluções produzidas pela regra LFM-LFJ, se os subconjuntos não forem encapsulados, também não são garantidamente óptimas.

Se se alterar a medida de eficiência e se considerar o problema $Pm||\sum C_j$, existe uma regra que produz soluções óptimas. Trata-se da regra SPT, *shortest processing time first*. Agendar de acordo com esta regra significa agendar primeiros as tarefas com menores tempos de processamento. Se o problema for $Pm||\sum w_j C_j$ pode ser utilizada a regra WSPT, *weighted shortest processing time*. Segundo esta regra, deve ser agendada em primeiro lugar a tarefa com o maior quociente w_j / p_j. A regra WSPT não garante uma solução óptima, mas é demonstrável a seguinte desigualdade:

$$\frac{\sum w_j C_j(WSPT)}{\sum w_j C_j(OPT)} < \frac{1}{2}\left(1+\sqrt{2}\right)$$

Este resultado garante que, mesmo no pior caso, o valor da solução WSPT não é superior em mais de 21% (aproximadamente) ao valor da solução óptima, o que torna a regra WSPT numa heurística interessante.

Para o problema $Pm|p_j = 1, M_j|\sum C_j$, a regra LFJ continua a produzir soluções óptimas, se os subconjuntos M_j forem encapsulados.

No entanto, o problema anterior é um caso particular do problema $Rm||\sum C_j$, se se considerar que o tempo de processamento de uma tarefa j numa máquina que não pertença ao conjunto M_j é arbitrariamente grande, tornando o agendamento da tarefa nessa máquina impossível. Este problema pode ser transformado num problema de emparelhamento com custos num grafo bipartido.

Se forem permitidas preempções, de uma maneira geral, os problemas mais fáceis de resolver. Para o problema $Pm|pmtn|C_{max}$, designando a tarefa longa por p_1, demonstra-se facilmente que

$$C^*_{max} = max\left(p_1, \frac{1}{m}\sum_{j=1}^{n} p_j \right)$$

Conhecendo à partida o valor da solução óptima, é relativamente fácil construir um algoritmo que determine o agendamento óptimo. Esse algoritmo, devido a McNaughton [13] , é composto por 3 passos:

1. Agendar as n tarefas numa única máquina e numa sequência arbitrária. O valor de C_{max} resultante deverá ser inferior ou igual a mC^*_{max} ;

2. Dividir este agendamento em m intervalos de tempo de tamanho igual a C^*_{max} ;

3. Cada intervalo de tempo conterá o agendamento de cada uma das máquinas.

Ainda para este problema há uma outra regra de agendamento que é óptima. Trata-se da regra LRPT, *longest remaining processing time first*. Esta regra é a versão preemptiva da regra LPT. A utilização desta regra, em geral, provoca um número de preempções infinito, o que, na prática, invalida a sua utilização. No entanto, este problema pode ser ultrapassado se se considerar que a máquina só pode ser interrompida em momentos de tempo discretos. O problema do número de preempções infinito é ultrapassado e demonstra-se que a regra continua a produzir soluções óptimas.

Generalizando ao ambiente $Qm|pmtn|C_{max}$ há a regra LRPT-FM, *longest processing time on the fastest machine*. A regra especifica que a tarefa cujo tempo processamento restante seja maior, deve ser agendada na máquina mais No limite, a regra LRPT-FM igualiza todos os tempos de processamento restantes e faz com que as tarefas mudem constantemente de máquina, provocando um número infinito de preempções. Embora produza soluções óptimas a regra LRPT-FM também não pode ser utilizada na prática. Mais

vez, impondo que as preempções só podem ocorrer em momentos resolve o problema: o número de preempções deixa de ser infinito e as produzidas são óptimas.

Se a regra LRPT-FM for aplicada às tarefas disponíveis a cada momento também se demonstra que a regra produz soluções óptimas no ambiente $Qm|r_j, pmtn|C_{max}$.

Considere-se agora o problema $Qm|pmtn|\sum C_j$. Este problema é uma generalização do ambiente com máquinas paralelas idênticas. A análise deste problema levou à formulação da regra SRPT-FM, *shortest remaining processing time on the fastest machine*. De acordo com esta regra, a tarefa com menor tempo de processamento restante deve ser colocada na máquina mais rápida, a segunda menor na segunda máquina mais rápida, etc. As preempções ocorrem quando a máquina mais rápida acaba o processamento, momento em que todas as tarefas em processamento são transferidas para a próxima máquina mais rápida, iniciando-se uma tarefa (se existir) na máquina mais lenta. Ou seja, a aplicação da regra conduz a um número de preempções finito. Também se demonstra que a regra SRPT-FM conduz a soluções óptimas.

Finalmente, os problemas com critérios de eficiência relacionados com as de conclusão. Considere-se o problema $Qm|pmtn|L_{max}$. Este problema é uma generalização do ambiente de máquinas idênticas. Pretende-se verificar se é possível agendar todas as tarefas com $L_{max} = z$, ou seja, qualquer momento conclusão C_j deve ser inferior ou igual a $d_j + z$, que funciona como uma
A resposta a esta questão pode ser dada resolvendo o problema
Imagine-se que a direcção do tempo foi invertida. Tomem-se as *deadlines* do problema original como datas de disponibilidade e aplique-se a regra LRPT- ao contrário, começando pela última *deadline*. Se todas as tarefas forem

depois do momento zero, significa que é possível fazer o agendamento com $L_{max} = z$. Para encontrar o valor óptimo de L_{max} basta uma pesquisa simples melhor valor de z.

A técnica das *deadlines* também pode ser utilizada no problema $Qm|r_j, pmtn|L_{max}$, ou seja cada data de conclusão d_j ser substituída por uma *deadline* $d_j + z$. Neste caso, inverter o problema não traz qualquer vantagem, pois resulta num problema do mesmo tipo, com datas de disponibilidade e datas de conclusão. Conforme se irá verificar na secção seguinte, este tipo de problemas pode ser transformado em problemas de maximização de fluxo numa rede.

O conjunto de exemplos apresentados nos parágrafos anteriores demonstra como a área do processamento paralelo é um campo de estudo vasto e aberto, uma vez que há uma grande variedade de situações que podem ocorrer: com/sem preempção, com/sem precedências, com/sem tempos de preparação de máquinas dependentes da sequência, processadores idênticos/uniformes/não relacionados, etc.

Devido a este facto apenas existem resultados para modelos bastante simples, que pouco têm a ver com o problema apresentado neste trabalho. No entanto, é a partir da análise de modelos simples que surgem ideias e princípios que podem ser aplicados em modelos mais complexos.

Nas duas secções que se seguem serão apresentados dois problemas bem conhecidos: o problema do fluxo máximo e o problema de transportes. Conforme se irá demonstrar, estes problemas podem ter grande aplicabilidade quando se trata de problemas com datas de disponibilidade e datas de conclusão.

4.3 O problema do fluxo máximo

A programação preemptiva de máquinas paralelas é um problema que, sob determinadas condições, pode ser formulado como um problema de maximização de fluxo numa rede direccionada. A seguinte formulação, como o exemplo, foram retirados de Ahuja et al. [1].

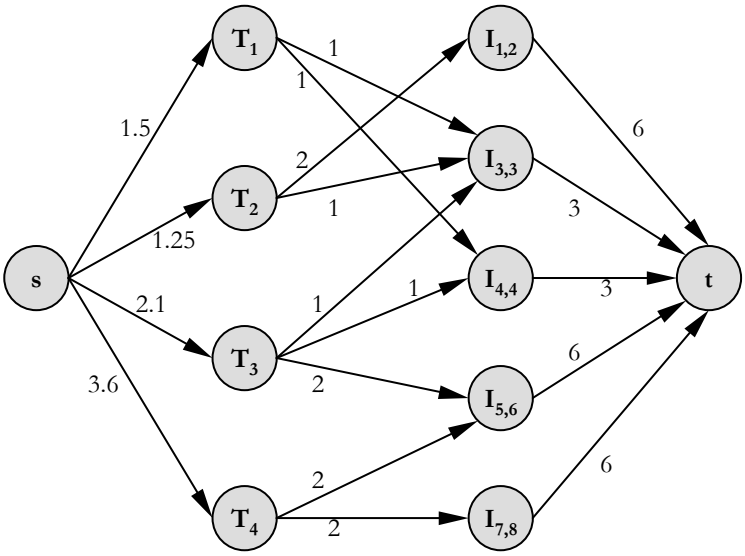
Considere-se o problema de programação $P|pmtn,r_j|\cdots$. Considere-se ainda a seguinte instância do referido problema:

Tarefa (T_j)	1	2	3	4
Duração (p_j)	1.5	1.25	2.1	3.6
Disponibilidade (r_j)	3	1	3	5
Conclusão (d_j)	5	4	7	9

O número de máquinas a programar é 3. As durações das tarefas são expressas em períodos-máquina necessários para a execução, podendo o período ter uma duração arbitrária. Admita-se ainda que as datas de disponibilidade se referem ao início do período em que a tarefa passa a estar disponível e que as datas de conclusão se referem ao início de período em que a tarefa deve estar concluída.

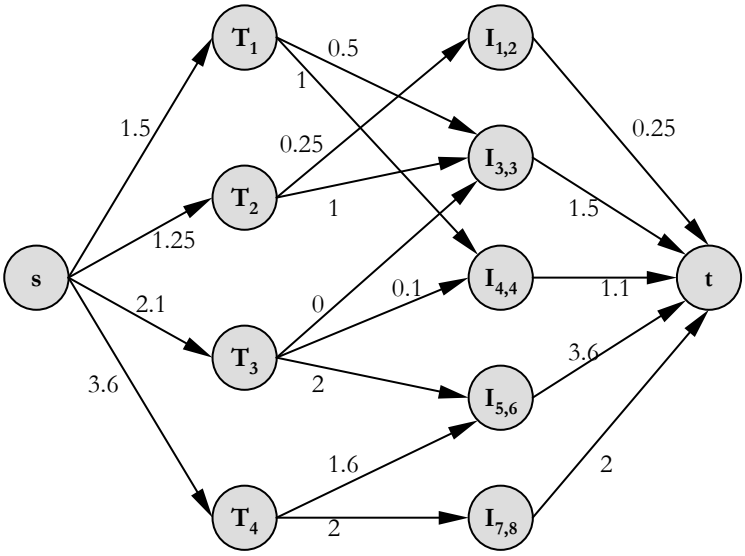
Na formulação como problema de fluxo máximo, começa-se por ordenar, forma crescente e em conjunto, as datas de disponibilidade e de conclusão. caso concreto essa ordenação produziria o resultado 1, 3, 4, 5, 7, 9. Com datas serão determinados $P \le 2 \cdot n - 1$ intervalos de tempo, que medeiam duas datas consecutivas. Seja $I_{k,l}$ o intervalo que vai entre o início do e o início do período $l+1$. Os intervalos para este problema serão $I_{1,2}$, $I_{3,3}$, $I_{5,6}$ e $I_{7,8}$. Repare-se que dentro de cada intervalo as tarefas disponíveis são mesmas, isto é, dentro de cada intervalo $I_{k,l}$ estão disponíveis as tarefas em $r_j \ge k$ e em que $d_j \le l+1$.

Pode-se agora construir um grafo bipartido para representar o problema em questão. Coloque-se no grafo um vértice de origem, s, e um vértice de destino, t, um vértice para cada uma das tarefas e um vértice para cada um dos intervalos construídos. Ligue-se a origem aos vértices das tarefas com arcos de capacidade igual a p_j, e ligue-se os vértices dos intervalos de tempo ao vértice de destino com arcos de capacidade $m \cdot (l - k + 1)$ (capacidade do conjunto de máquinas no intervalo). Finalmente, deve ligar-se cada vértice de tarefa aos vértices de intervalo em que essa tarefa esteja disponível com um arco que represente o número de períodos que se podem atribuir à tarefa naquele intervalo, que será sempre $l - k + 1$. Se este procedimento fosse seguido para o exemplo em análise, resultaria o seguinte grafo:



Depois de fazer a maximização do fluxo, a solução deve ser analisada. A solução de fluxo máximo só é uma solução de agendamento válida se o fluxo máximo for igual a $\sum_{j=1}^{n} p_j$. Se o fluxo máximo for menor, significa que nem todas as tarefas foram agendadas na totalidade, o que não é uma solução válida para o problema.

A construção de uma solução de agendamento a partir dos resultados do problema do fluxo máximo é trivial. A seguir mostra-se o grafo correspondente ao problema onde já se inscreveram os fluxos da solução óptima. O valor do fluxo máximo é igual a $\sum_{j=1}^{n} p_j$, pelo que a solução é válida.



Como já se afirmou, a solução pode ser facilmente transposta para um diagrama de Gantt.

Máquina	Período								
	1	2	3	4	5	6	7	8	9
1				1					
2			2		3				
3						4			

As máquinas utilizadas foram escolhidas arbitrariamente. Por exemplo, a tarefa 4 poderia ser realizada na máquina 1. Isto ilustra o tipo de resultado que é fornecido pelo problema de fluxo máximo: em vez de uma solução de agendamento, o problema de fluxo máximo permite verificar se o agendamento, dadas as restrições, é possível. No caso afirmativo, as diversas soluções alternativas fornecidas pelo problema de fluxo máximo são válidas.

4.4 O problema de transportes

Nesta secção será feita uma revisão ao problema de transportes, também conhecido como modelo de Hitchcock. Como se trata de um problema sobejamente conhecido, apenas será feita a formulação como caso particular do problema de fluxo de custo mínimo definido sobre um grafo bipartido e serão apresentados, de forma breve, os algoritmos a utilizar nos testes computacionais. Esta secção segue de perto Ahuja et al. [1] , onde poderão ser encontrados mais detalhes sobre o problema.

4.4.1 Enunciado do problema

O enunciado clássico do problema de transportes é o seguinte: determinada empresa possui p fábricas, cujas ofertas são conhecidas e q pontos de venda, cujas procuras são igualmente conhecidas. Qualquer fábrica pode fornecer qualquer ponto de venda, mas existe uma matriz de custos de transporte por unidade transportada entre cada par fábrica/ponto de venda. Pretende-se determinar que fábricas devem fornecer que pontos de venda e em que quantidades, de modo a satisfazer a procura nos pontos de venda e a minimizar os custos de transporte.

4.4.2 Formulação como problema de fluxo de custo mínimo

Seja $G = (N, A)$ um grafo orientado. Cada arco $(i, j) \in A$ tem associado um custo c_{ij} e uma capacidade u_{ij}. Cada vértice $i \in N$ está associado a uma quantidade $b(i)$ que indica a oferta, $b(i) > 0$, ou a procura, $b(i) < 0$, nesse vértice. Denotando o fluxo no arco $(i, j) \in A$ por x_{ij}, o problema de fluxo de custo mínimo pode ser formulado da seguinte forma:

Minimizar $\sum_{(i,j)\in A} c_{ij} x_{ij}$

sa

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \textit{para todo o } i \in N$$

$$0 \le x_{ij} \le u_{ij} \qquad \textit{para todo o } (i,j) \in A$$

O primeiro conjunto de restrições são normalmente designadas por equações de conservação de fluxo, pois garante que o fluxo que sai de um qualquer vértice menos o fluxo que entra nesse vértice é igual à oferta dessa vértice (ou procura, se a diferença for negativa). O segundo conjunto de restrições garante que as capacidades dos arcos não são excedidas e que não há fluxos negativos.

É usual associar-se ao problema um conjunto de pressupostos normalizadores, que não implicam qualquer perda de generalidade:

– Não há limites inferiores de capacidade nos arcos (o limite é zero para todos os arcos);

– Todos os dados são valores racionais;

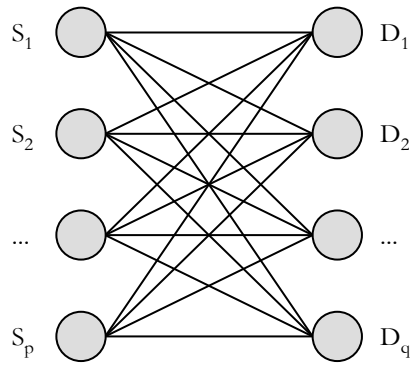
– O grafo é orientado (mesmo que não seja orientado, pode sempre ser transformado);

– A procura/oferta nos vértices satisfaz a condição $\sum_{i\in N} b(i) = 0$ e o problema possui pelo menos uma solução admissível;

– Todos os custos nos arcos são não negativos.

O problema de transportes pode ser formulado como um problema de fluxo de custo mínimo e resolvido como tal, recorrendo a algoritmos algo sofisticados que serão abordados na próxima secção.

O grafo que se segue representa o problema de transportes, formulado problema de fluxo de custo mínimo.



As origens (fábricas) são representadas por $S_1, S_2, \cdots S_p$ e os destinos (pontos de venda) por $D_1, D_2, \cdots D_q$. Considere-se que todos os arcos estão orientados das origens para os destinos. A soma da oferta no conjunto das origens é igual à soma da procura no conjunto dos destinos (se assim não acontecer no problema original, podem sempre ser criadas origens ou destinos fictícios). A cada arco está associado um determinado custo e uma capacidade infinita.

Formulando o problema de transportes desta forma são cumpridos todos os pressupostos enunciados para o problema de fluxo de custo mínimo.

4.4.3 Interesse do problema de transportes

O problema de transportes pode ser utilizado para resolver alguns problemas de agendamento directamente (poucos) e para obter limites para procedimentos de *branch-and-bound*. O exemplo que se apresenta a seguir foi retirado de Pinedo [16].

Considere-se o problema $Qm|p_j = 1|\sum C_j$. Se a tarefa T_j for agendada no processador P_i na k-ésima posição incorrer-se-á numa contribuição para o valor da função objectivo igual a k/b_i. Cada tarefa pode ser agendada nos vários processadores e nas várias posições, havendo $n \times m$ possibilidades de agendar cada tarefa

Se se considerar que cada tarefa corresponde a uma origem de oferta igual a 1 (duração da tarefa) e que cada possibilidade de agendamento ($n \times m$ possibilidades) constitui um destino de procura também igual a 1, o problema pode ser resolvido aplicando os algoritmos do problema de transportes.

O modelo que será desenvolvido no próximo capítulo irá utilizar uma formulação algo diferente desta, mas baseada no problema de transportes.

4.4.4 Algoritmos

No presente trabalho será utilizada uma formulação baseada no problema de transportes, que será descrita no próximo capítulo. Para a realização de testes computacionais tornou-se necessário o recurso a um código para resolver eficientemente instâncias grandes do problema de transportes. Esse código é designado por RELAX-IV. Trata-se de um código de domínio público e as ideias que lhe estão subjacentes encontram-se numa publicação de Bertsekas [2].

Refira-se que se trata de um código para a resolução de problemas de fluxo de custo mínimo, do qual o problema de transportes, como já se referiu, é um caso particular, e baseia-se num algoritmo primal-dual e em algumas estratégias de relaxação.

De resto, o código é instrumental em relação ao presente trabalho, isto é, apenas será utilizado para resolver instâncias do problema de transportes. Como o código será tratado como uma caixa negra, não faz muito sentido entrar em pormenores sobre o referido código ou sobre os algoritmos que lhe estão subjacentes, até porque não teria qualquer interesse para o presente trabalho.

4.5 Conclusão

Neste capítulo, começou-se por fazer uma caracterização genérica dos de planeamento operacional e por introduzir notação matemática capaz de

descrever eficientemente os diversos tipos de problemas. De seguida fez-se uma incursão no assunto que constitui o tema central deste trabalho: o processamento paralelo. Introduziram-se alguns problemas já estudados e para os quais existem resultados. Desses problemas ficam duas ideias fundamentais:
– O objecto deste trabalho é muito mais complexo que esses problemas: quando muito, os problemas apresentados, à semelhança de outros que se poderiam referir, fornecerão ideias e pistas para o problema em análise;

– Há uma grande variedade de situações que podem ocorrer em problemas de processamento paralelo: possivelmente, qualquer processo de resolução que possa ser desenvolvido, contemplará seriamente apenas as situações mais prementes, deixando outras situações numa posição secundária.

A primeira observação justifica o presente trabalho: não é conhecida nenhuma forma de lidar simultaneamente com o conjunto de situações colocadas pelo problema em análise. Encontram-se na literatura muitas ideias interessantes, mas que, por um ou outro motivo, não podem ser aplicadas directamente.

A segunda observação reflecte um pouco o tipo de trabalho que poderá ser realizado: análise do sistema de modo a distinguir o que é realmente importante do que é acessório e busca de soluções para as questões mais importantes, eventualmente contemplando as questões menos importantes.

Finalmente, são revistos o problema do fluxo máximo e o problema de transportes, devido à grande utilidade destes para o presente trabalho.

Capítulo 5
ABORDAGEM DO PROBLEMA

Uma vez apresentada a literatura existente que servirá de base ao desenvolvimento de um modelo de resolução para o problema em análise, é momento de fazer a formulação matemática dos vários aspectos do

de propor um modelo para a resolução do problema. Este capítulo irá versar sobre estas questões e, juntamente com o capítulo seguinte, pode ser considerado o núcleo do presente trabalho.
Na primeira secção será feita uma pequena introdução à estratégia de abordagem ao problema, onde serão referidas as principais dificuldades do próprio problema. Fundamentalmente, pretende-se introduzir e justificar a filosofia que será seguida ao longo do capítulo.

Na segunda secção, começar-se-á por recapitular todas as condicionantes do problema em análise. Embora estas já tenham sido referidas no capítulo 3, acompanham-se agora com a notação matemática introduzida na revisão de literatura. Segue-se a formulação de um modelo de programação matemática genérico, onde se incluirão todas as restrições do problema.

Na terceira secção demonstrar-se-á que devido, entre outras coisas, à grande complexidade do problema em análise, a concretização do modelo genérico apresentado na primeira secção seria extremamente difícil. Em alternativa, proposta uma abordagem em que algumas restrições não são introduzidas, tornando o problema praticável. Esta abordagem será composta por um algoritmo principal, que fornecerá soluções que podem ser depois por heurísticas bastante simples. A solução final, resultante da aplicação do algoritmo e das heurísticas (apresentadas no capítulo seguinte), não é garantidamente óptima.

5.1 Introdução

A abordagem de um problema deste tipo é uma tarefa complexa e difícil, necessariamente composta de várias etapas. A primeira etapa, tal como em qualquer problema de investigação operacional, é a observação detalhada do sistema, com vista a estabelecer relações entre os seus vários componentes.

Por norma, trata-se de sistemas com elevada complexidade, tal como o sistema em análise. Assim, torna-se necessário delimitar o objecto da análise. No caso concreto, ao ficar evidente que a secção de tricotagem era o "gargalo" do sistema, o resto do sistema foi ignorado e a atenção concentrou-se naquela secção. Identificou-se assim um problema de agendamento de máquinas paralelas e identificaram-se as respectivas características.
Se ainda assim o problema permanecer excessivamente complexo para ser objecto de resolução exacta, é necessário encontrar uma formulação que, sem descaracterizar o problema, diminua a complexidade e permita encontrar soluções de boa qualidade. Foi esta a estratégia seguida na abordagem ao problema concreto deste trabalho.

Uma vez encontrada uma formulação que produza soluções, é necessário para estas soluções de modo a identificar as principais lacunas e os pontos podem ser melhorados. Os resultados desta observação podem ser identificar deficiências na formulação utilizada ou para desenvolver que possam melhorar as soluções obtidas. No caso concreto do problema estudado, foi possível identificar melhorias que poderiam ser feitas nas produzidas pelo modelo. Foram então desenvolvidos algoritmos capazes de introduzir melhorias naquelas soluções.

Desta forma, seguindo esta filosofia, foi possível construir um modelo que produz soluções para um problema de grande complexidade, considerando o maior número possível de restrições do problema original.

5.2 Formulação matemática do problema

5.2.1 Decomposição do problema

A notação que será utilizada ao longo do capítulo é a notação já introduzida na revisão de literatura.

5.2.1.1 Processadores

Seja P o conjunto dos m processadores que compõem o sistema, tal que, $P = \{P_1, P_2, \cdots P_m\}$.
Embora o conjunto de processadores do problema concreto tenha velocidades semelhantes que, na prática, sem grandes desvios, poderiam ser consideradas iguais, tornando os processadores idênticos, vamos considerar, generalizando, que os processadores são uniformes: a cada processador corresponde uma constante b_i que designa a velocidade de processamento, que é independente da tarefa executada.

Esta constante, b_i, é na realidade um factor que deverá ser igual a 1 num processador que funcione a uma velocidade padrão arbitrária. Será maior que 1 no caso de processadores mais rápidos que o padrão e será inferior a 1 para processadores mais lentos que o padrão.

5.2.1.2 Tarefas

Seja T o conjunto das n tarefas a agendar, tal que $T = \{T_1, T_2 \cdots T_n\}$.
A cada tarefa está associado o seguinte conjunto de constantes:

- Tempos de processamento;
- Data de disponibilidade;
- Data de conclusão;
- *Deadline*;
- Peso (importância).

A cada tarefa está associado um vector de tempos de processamento, representando cada elemento p_{ij} desse vector, o tempo que o demora a processar a tarefa T_j. Como os processadores são $p_{ij} = p_j / b_i$, representando p_j o tempo de processamento no padrão.

Devido à estrutura do modelo que será introduzido adiante e, uma vez que as diversas ordens se referem a produtos homogéneos, é de toda a conveniência que a duração de uma tarefa possa ser expressa em unidades a produzir. Neste caso concreto, a unidade seria o par de peúgas. A notação que se apresenta de seguida é em tudo equivalente à que foi referida no parágrafo anterior.
Considere-se que, à semelhança do que acontece com a velocidade processadores, existe um tempo de execução padrão para 1 unidade Mais, há produtos em que 1 unidade demora mais que o tempo produzida, e há produtos em que essa mesma unidade demora tempo padrão a ser produzida. Seja c_j um factor que reflicta o tempo de 1 unidade de produto da tarefa T_j, tal que, se o tempo de ao tempo padrão, $c_j = 1$ e, se o tempo de execução for superior ou inferior ao tempo padrão, c_j seja superior ou inferior a 1, respectivamente.

Designe-se por p_u o tempo execução padrão de uma unidade de produto. Se este tempo for medido no processador padrão, o cálculo do tempo de processamento unitário para uma determinada tarefa num determinado processador, p_{uij}, pode ser efectuado pela seguinte expressão:

$$p_{uij} = \frac{p_u \cdot c_j}{b_i}$$

Exemplo

Considere-se uma situação de 3 processadores e 3 tarefas: para os processadores, b_1, b_2 e b_3 valem, respectivamente, 1.1, 1.0 e 0.9; paras as tarefas, c_1, c_2 e c_3 valem, respectivamente, 0.8, 1.0 e 1.2 e devem ser produzidas 10 unidades em cada tarefa; o tempo de processamento unitário padrão, medido no processador padrão é de 1 minuto (60 segundos).

Aplicando a expressão anterior, pode verificar-se que, se a tarefa 1 fosse integralmente realizada no processador 1, demoraria 43.6 segundos ((60)(0.8)/(1.1)) por unidade produzida, ou seja, 436 segundos para a das 10 unidades. Caso a tarefa 2 fosse produzida no processador 2 o tempo unitário de processamento seria 60 segundos ((60)(1.0)/(1.0)). Caso a tarefa fosse produzida no processador 3 o tempo de processamento unitário seria segundos ((60)(1.2)/(0.9)). Desta forma, poderiam ser calculados os tempos de processamento referidos inicialmente, o que torna as duas equivalentes. ■

Para evitar ambiguidades, sempre que a duração de uma tarefa esteja expressa em unidades a produzir, ela será referenciada como p'_j. A expressão a utilizar no cálculo do vector de tempos de processamento, que faz com que as notações sejam equivalentes, será

$$p_{ij} = p'_j \cdot \frac{p_u \cdot c_j}{b_i}$$

Saliente-se que é necessário assegurar a consistência entre os coeficientes b_i e os coeficientes c_j. Essa consistência é assegurada se p_u significar "o tempo que leva a produzir 1 unidade de produto padrão, num processador padrão".

Quanto às datas de disponibilidade, datas de conclusão, *deadlines* e pesos, os valores têm o significado que lhes foi dado na revisão de literatura.

As tarefas admitem preempção e podem ser processadas, simultaneamente, em vários processadores, relaxando uma das condições que aparecem na generalidade dos modelos de planeamento operacional e que foi referida na revisão de literatura (a condição de não existência de processamento simultâneo).

Não existem quaisquer relações de precedência entre as tarefas. O que existe questão dos sortidos, ou seja, é de toda a conveniência que tarefas ao mesmo sortido sejam executadas de forma mais ou menos simultânea.

5.2.1.3 Matriz de adequação processador/tarefa

Como foi referido, determinada tarefa deve ser realizada, determinados processadores. Isto implica penalizar soluções em que preferências não sejam respeitadas. O problema deverá então ter coeficientes de adequação. Designar-se-ão esses coeficientes por e_{ij}, representando cada um a penalidade associada a agendar a tarefa T_j no processador P_i.

Os coeficientes desta matriz, e_{ij}, podem ser preenchidos de duas formas diferentes:

– Com um conjunto de valores que representem os diversos níveis de adequação possíveis entre máquinas e produtos, sendo estes valores substituídos, mais tarde, por constantes que representem a penalidade associada a cada nível, podendo existir um número arbitrário de níveis;

– Com valores arbitrários de penalidades;

A primeira hipótese parece mais correcta, pois, além de vantagens computacionais, permite manter os valores das penalidades como parâmetros de afinação do modelo.

Sobre a escolha das máquinas mais adequadas, embora sem estar relacionado com o exposto atrás, há a questão da adjacência das máquinas a utilizar em cada tarefa, ou seja, uma tarefa deve ser agendada em máquinas o mais próximas possível umas das outras.

5.2.1.4 Matriz de tempos de preparação

Como os tempos de preparação dos processadores são dependentes de tarefas agendadas, existe uma matriz de tempos de preparação preenchida com os tempos de preparação de uma máquina na T_{j_1} para a tarefa T_{j_2}, ou seja, os coeficientes $s_{j_1 j_2}$.

É razoável admitir que os tempos de preparação devem ser maiores ou iguais que zero e que devem satisfazer a condição conhecida como "desigualdade triangular", ou seja, dadas três quaisquer tarefas a, b e c, a proposição $s_{ab} + s_{bc} \geq s_{ac}$ deve ser sempre verdadeira. A inclusão de uma linha adicional na matriz, com os coeficientes s_{0j}, que represente a passagem do estado inicial do sistema para uma qualquer tarefa, também é aceitável. Para uma tarefa que esteja em curso (ou outra que exija a mesma preparação), este coeficiente seria nulo.

5.2.1.5 Estado inicial

Como já foi referido, o sistema é caracterizado por um estado inicial, ou seja, em determinado momento, qualquer máquina está com uma determinada preparação e pode estar a produzir ou estar parada (por opção de agendamento ou por avaria). No caso de estar a produzir, pode não ser conveniente a preempção da tarefa em curso. Isto implica que a máquina pode não estar disponível imediatamente. Pode então associar-se a cada máquina uma data de disponibilidade a_i.

5.2.2 Formulação de um modelo de programação matemática

Nesta secção será feita a formulação de um modelo de decisão típico de programação matemática, em que se pretende minimizar uma determinada função objectivo através do ajuste de variáveis de decisão, sujeitas a um conjunto de restrições.

5.2.2.1 Variáveis de decisão

Uma formulação possível para este tipo de problemas, consiste em tempo de agendamento disponível em cada processador, em fragmentos de tamanho mais ou menos arbitrário. Em cada um fragmentos temporais pode, em princípio, ser agendado um dimensão de uma qualquer tarefa. Seja U o tamanho desse

Isto implica desde logo que o tamanho do fragmento deve ser um divisor comum a todas as tarefas. A unidade de produto é um bom divisor se as quantidades envolvidas forem suficientemente pequenas para o permitir. Caso contrário deverá ser encontrada uma outra dimensão para os fragmentos temporais.

Quanto mais pequeno for o tamanho de fragmento, maior será a precisão do agendamento, mas, em contrapartida, os recursos computacionais necessários à resolução do problema serão maiores. O inverso é verdadeiro se os fragmentos forem maiores, ou seja, o problema será resolvido com menor precisão, exigindo, em contrapartida, menos recursos computacionais.

No caso concreto da empresa em análise, ficou explícito que o fragmento de agendamento poderia ir, sem qualquer problema, até às 8 horas, num sistema onde a produção por hora e por processador é de cerca de 8 unidades de produto. Faz sentido pensar que, quanto maiores as quantidades envolvidas no agendamento, menor a precisão necessária, fazendo com que os fragmentos de tempo a considerar no agendamento possam ser maiores.

Cada fragmento de tempo disponível estaria associado a uma variável de decisão binária. Designe-se por K o número de fragmentos que constituem o horizonte de agendamento. As variáveis de decisão seriam x_{ijk}, com $i = 1,2,\cdots,m$, com $j = 1,2,\cdots,n$ e com $k = 1,2,\cdots,K$. A variável x_{ijk} tomaria o valor de 1 se um fragmento da tarefa T_j estivesse agendada no processador P_i no fragmento k. Caso contrário a variável tomaria o valor zero.

Caso se tratasse de processadores idênticos a explicação ficaria por entanto, o problema que se pretende modelar é composto por uniformes. A questão coloca-se nos seguintes termos: para um horizonte de agendamento, os processadores mais rápidos poderão

maior número de fragmentos de tarefa que os processadores mais determinar o valor de K?

Seja K_i o número de fragmentos que o processador P_i consegue processar no horizonte de agendamento H. Os valores de K_i podem ser calculados pela seguinte expressão:

$$K_i = \left\lfloor \frac{H}{U} \cdot b_i \right\rfloor$$

Para que se possa proceder ao agendamento de todos os processadores no horizonte de tempo escolhido, K deverá ter um valor igual ao número de fragmentos que é possível processar no processador mais rápido, ou seja,

$$K = max(K_i)$$

Os diversos valores de K_i constituirão os limites ao agendamento dos diversos processadores, ou seja, devem ser acrescentadas restrições ao problema de modo a evitar que, em cada processador, haja agendamento de fragmentos de tarefas entre o período $K_i + 1$ e o período K.

Exemplo

Como exemplo, considerem-se 4 processadores com coeficientes b_i iguais a 1.2, 1.0, 1.1 e 0.8. Pretende-se fazer o agendamento para 10 dias e o fragmento de agendamento é o dia. Num processador padrão, existirão 10 períodos de agendamento. O processador 1, o mais rápido, com $b_1 = 1.2$, consegue processar o equivalente a 12 horas de processador padrão, ou seja, $K = K_1 = 12$. No diagrama seguinte são resumidas as possibilidades de agendamento para os 4 processadores:

Máquina	Período											
	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												

A sombreado estão representados os períodos onde não é fisicamente possível agendar, pois correspondem a períodos para lá do horizonte de agendamento. Como já foi afirmado, esta questão deverá ser tratada a nível de restrições. ■

Para finalizar, as grandes vantagens desta estrutura de dados são a sua fácil compreensão e, simultaneamente, a sua grande potência: com esta abordagem pode obter-se a precisão de agendamento desejada, desde que haja recursos computacionais, através da redução do tamanho do período de agendamento.

5.2.2.2 Função objectivo

A função objectivo escolhida, é uma função de minimização de penalidades resultantes do agendamento. Tal como se irá verificar, trata-se de uma função objectivo não linear.

As penalidades que se irão introduzir na função objectivo resultam da análise do problema e são de cinco tipos:

– Penalidades por violação de datas de conclusão: este tipo de penalidade pode ser concretizado através de vários tipos de funções e visa penalizar o agendamento fora da janela temporal válida. A penalidade pode crescer linearmente com a dimensão da violação ou pode crescer exponencialmente. Se, adicionalmente, existirem *deadlines*, estas terão que ser tratadas como restrições, de modo a que o agendamento fora das datas impostas pelas *deadlines* seja inviabilizado;

– Penalidades por inadequação máquina/produto: como já foi referido, é conveniente agendar as encomendas nas máquinas que as

produzam com maior índice de qualidade. Este tipo de penalidade é uma função linear, pois resulta da multiplicação das variáveis de decisão pelos valores de penalidade derivados da matriz de adequações produto/máquina;
– Penalidades por tempos de preparação: esta penalidade deve ser uma função linear do tempo de preparação em que se incorre;

– Penalidades devido aos sortidos: sempre que houver sortidos, deve haver penalidades para o desfasamento de ordens de fabrico de um mesmo sortido;

– Penalidades devido à localização física das máquinas utilizadas: para facilitar o trabalho dos operadores do parque de máquinas e dos AGV, uma encomenda deve ser executada em máquinas o mais próximas possível entre si. Sempre que isto não aconteça deve existir uma penalidade pela dispersão.

A função objectivo genérica será a soma de cinco funções que representem adequadamente os cinco tipos de penalidades apresentados. Designando essas funções por f_1, f_2, f_3, f_4 e por f_5, correspondendo à ordem pela qual foram apresentadas, a função objectivo será:

$$min \quad f_1 + f_2 + f_3 + f_4 + f_5$$

em que:

$$f_1 = f\left(x_{ijk}, d_j\right),$$

$$f_2 = f\left(x_{ijk}, e_{ij}\right),$$

$$f_3 = f\left(x_{ijk}, s_{j_1 j_2}\right),$$
$$f_4 = f\left(x_{ijk}, \cdots\right),$$

representando as reticências a informação sobre sortidos,
$f_5 = f(x_{ijk}, \cdots)$,

em que a informação sobre a disposição física das máquinas é representada pelas reticências.

Para controlar o peso que cada tipo de penalidade tem na função objectivo, as respectivas funções deveriam ser multiplicadas por constantes cujos valores são arbitrários e devem reflectir o peso que se atribui a cada componente da penalidade total, ou seja, a função objectivo ficaria

$$min \quad w_1 \cdot f_1 + w_2 \cdot f_2 + w_3 \cdot f_3 + w_4 \cdot f_4 + w_5 \cdot f_5$$

5.2.2.3 Restrições

Para garantir a integridade do modelo é necessário estabelecer desde logo uma série de restrições ligadas à definição das variáveis de decisão utilizadas:

$$\sum_{j=1}^{n} x_{ijk} \leq 1 \quad \text{para } i = 1,2,\cdots,m \text{ e para } k = 1,2,\cdots,K$$

$$\sum_{i=1}^{m}\sum_{k=1}^{K} x_{ijk} \cdot U = p_j \quad \text{para } j = 1,2,\cdots,n$$

$$\sum_{j=1}^{n}\sum_{k=K_i+1}^{K} x_{ijk} = 0 \quad \text{para } i = 1,2,\cdots,m$$

$$x_{ijk} \geq 0 \text{ e inteiro} \quad \text{para } i = 1,2,\cdots,m,\ j = 1,2,\cdots,n \text{ e } k = 1,2,\cdots,K$$

O primeiro conjunto de restrições garante que em qualquer máquina e em qualquer período de tempo estará agendada, no máximo, apenas uma tarefa. O segundo conjunto de restrições garante que cada tarefa é agendada na totalidade e apenas na totalidade. A terceiro conjunto de restrições evita que sejam agendadas tarefas nos

processadores mais lentos para além do horizonte de agendamento e resulta da definição do valor de K. O quarto conjunto de restrições condições de não negatividade das variáveis de decisão.

Deadlines e datas de disponibilidade das tarefas

Se existirem *deadlines* e datas de disponibilidade é necessário incluir restrições adicionais. Considere-se que, para qualquer tarefa T_j, a condição $\widetilde{d}_j \geq d_j$ é verdadeira, isto é, se existir uma *deadline*, esta é sempre igual ou posterior à data de conclusão. As restrições a acrescentar são as seguintes:

$$\sum_{i=1}^{m}\sum_{k=1}^{r_{ij}-1} x_{ijk} = 0 \qquad \text{para } j = 1,2,\cdots,n$$

$$\sum_{i=1}^{m}\sum_{k=\widetilde{d}_{ij}+1}^{K} x_{ijk} = 0 \qquad \text{para } j = 1,2,\cdots,n$$

O primeiro conjunto de restrições não permite o agendamento de qualquer tarefa antes do período r_{ij}. O segundo conjunto de restrições não permite o agendamento depois do período $\widetilde{d}_{ij}$. Os valores de r_{ij} e de $\widetilde{d}_{ij}$ devem ser definidos em relação a cada processador em concreto, tendo em conta a sua velocidade de processamento:

$$r_{ij} = \lfloor r_j \cdot b_i \rfloor$$

$$\widetilde{d}_{ij} = \lfloor \widetilde{d}_j \cdot b_i \rfloor$$

O que as expressões anteriores significam é algo como o seguinte: se num determinado processador uma determinada data se referir ao período k, num processador a funcionar ao dobro da velocidade, essa data referir-se-á ao período $2k$. Esta conclusão aparece na

sequência do que já foi discutido na definição das variáveis de decisão.

Incompatibilidades tarefa/processador

As incompatibilidades entre produtos e máquinas também devem ser tratadas ao nível das restrições. Assim, para cada par produto/máquina que sejam incompatíveis, deverá ser acrescentada a seguinte restrição:

$$\sum_{k=1}^{K} x_{ijk} = 0 \quad \text{para cada par } P_i, T_j \text{ incompatível}$$

Estas restrições asseguram que, se uma tarefa for incompatível com determinada máquina, essa tarefa não será agendada em nenhum período nessa mesma máquina.

Paragens programadas de processadores

Se for útil a introdução de períodos de paragem programada no sistema, isso pode ser feito a nível de restrições. Admita-se que há um conjunto, S, de paragens programadas a introduzir no modelo. Cada elemento de S tem o formato (N, k_b, k_e), onde N é o subconjunto de processadores que devem parar no período $[k_b, k_e]$. O seguinte conjunto de restrições deve ser acrescentado:

$$\sum_{j=1}^{n} \sum_{k=k_{bi}}^{k_{ei}} x_{ijk} = 0 \quad \text{para todo o } i \in N \text{ e para todo o } (N, k_b, k_e) \in S$$

Tal como foi feito para as datas de disponibilidade e de conclusão, e pelo mesmo motivo, para cada elemento de S devem ser calculados os valores k_{bi} e k_{ei} relativos a cada elemento de N. As expressões são as seguintes:

$$k_{bi} = \lfloor k_b \cdot b_i \rfloor$$

$$k_{ei} = \lfloor k_e \cdot b_i \rfloor$$

Datas de disponibilidade dos processadores
Finalmente, no caso de os processadores, por qualquer razão, possuírem uma data de disponibilidade, a_i, e não se deseje agendar nenhuma tarefa no período $[1, a_i - 1]$ deve ser acrescentado o seguinte conjunto de restrições:

$$\sum_{j=1}^{n} \sum_{k=1}^{a_i - 1} x_{ijk} = 0 \quad \text{para todo o } i$$

Neste caso pode considerar-se que a_i já está referido em períodos de agendamento de um determinado processador, não sendo por isso necessário fazer a sua conversão.

Este modelo aborda todas as questões do problema, excepto uma questão relacionada com os tempos de preparação das máquinas. A questão que se coloca é a de assegurar, ao nível das restrições, que a adição dos tempos de preparação dos processadores não provoca violações nas *deadlines* ou aumentos nas violações às datas de conclusão. Efectivamente, pode acontecer que, depois de adicionar ao agendamento os tempos de preparação nas mudanças de tarefa, as *deadlines* sejam violadas ou haja aumentos nas violações às datas de conclusão.

No entanto, não existe nenhuma forma óbvia de contemplar a questão a nível de restrições. Uma solução para contornar o problema reside no seguinte: pelas leis estatísticas, os tempos de preparação distribuir-se-ão pelos processadores de forma homogénea. Para compensar estes tempos poderão ser colocadas folgas no agendamento. A forma de introduzir estas folgas podem ser diversas. Por exemplo, conhecendo a percentagem de tempo que é ocupada pela preparação das máquinas, poder-se-ia inflacionar os tempos de execução das tarefas nessa percentagem. Desse modo, como os tempos reais iriam ser menores, compensariam o tempo despendido na preparação das máquinas. Outra forma de contornar o problema seria ignorá-lo devido à insignificância dos tempos envolvidos. Este último procedimento poderá ser adoptado no problema em análise pois os tempos de preparação são da ordem de 0.1‰ do tempo de execução das tarefas.

5.2.3 Limitações do modelo genérico apresentado

O modelo introduzido na secção anterior pretendeu ser uma introdução ao modelo que será apresentado nas secções que se seguem, e que será baseado no problema de transportes. A vantagem de o fazer reside nas semelhanças entre a estrutura dos dois modelos, facilitando a apresentação do modelo definitivo.
Embora este modelo tenha utilidade do ponto de vista conceptual, uma vez que contempla a quase totalidade das questões envolvidas, a sua implementação seria muito difícil.

Desde logo, seria um modelo de dimensões gigantescas, exigindo uma considerável capacidade computacional para a sua resolução. Por um modelo com 50 máquinas, 50 tarefas e um horizonte temporal de 50 períodos, que é um modelo relativamente pequeno, haveria, no mínimo, coisa como 127,700 restrições. O número de variáveis de decisão seria de 125,000.

Outra grande dificuldade seria definir a função objectivo. As funções de penalização por violação de janelas temporais e de incompatibilidades processador/tarefa são relativamente triviais. As outras funções de penalização (tempos de preparação, sortidos e localização física) seriam de definição complexa. A partir das variáveis de decisão definidas, seria muito difícil encontrar expressões para o cálculo dos tempos de preparação, das penalidades associadas ao desfasamento entre tarefas do mesmo sortido, ou das penalidades associadas à utilização de máquinas fisicamente distantes para a execução da mesma tarefa.

Assim, torna-se necessário encontrar uma formulação em que algumas destas condicionantes sejam relegadas para segundo plano, diminuindo a complexidade do modelo sem comprometer a obtenção de soluções razoáveis. Um modelo deste tipo será apresentado na próxima secção.

5.3 Formulação de um modelo de resolução

Na presente secção irá ser formulado e explicado o modelo que se propõe para a abordagem ao problema em análise. Conforme se irá ver, trata-se de resolver uma relaxação do problema, recorrendo a uma formulação baseada no problema de transportes. A solução assim encontrada será depois melhorada por heurísticas muito simples, especificamente desenhadas para o efeito.
O modelo que se propõe é uma relaxação do modelo genérico introduzido na secção anterior, onde, numa primeira fase, é ignorada toda a informação relativa a tempos de preparação de máquinas, sortidos e localização física dos processadores a utilizar. Todas as outras restrições são incluídas neste modelo.

Como já foi referido, a obtenção de uma solução para esta relaxação será conseguida através da resolução de um problema de transportes. De seguida descrito como se faz a transformação do problema em análise num transportes.

5.3.1 Introdução

Conforme se demonstrou nas secções anteriores, a duração de uma qualquer tarefa, T_j, pode ser expressa em termos de unidades a produzir, sendo essa quantidade designada por p'_j.

Também foi referido que o tempo de produção de uma unidade de produto não é igual para todas as tarefas, existindo tarefas mais demoradas que o padrão e tarefas menos demoradas que o padrão. Essas diferenças são expressas no coeficiente de duração da tarefa, c_j. A partir daqui, a duração de uma tarefa pode ser expressa em unidades padrão a produzir. Designe-se essa quantidade por p'_{sj}, tal que,

$$p'_{sj} = p'_j \cdot c_j$$

No modelo genérico apresentado na secção anterior, o horizonte de agendamento, H, era dividido em fragmentos de tamanho U. Embora

valores tivessem sido apresentados como quantidades de tempo no processador padrão, também podem ser referidos como quantidades de unidades padrão a produzir no processador padrão. Designem-se essas quantidades por H' e por U'. A conversão entre uns valores e outros pode ser feita da seguinte forma:

$H' = H / p_u$ $\qquad\qquad$ $U' = U / p_u$

Recorde-se que p_u é o tempo de produção de 1 unidade padrão num processador a funcionar à velocidade padrão.

Desta forma, há uma unidade comum a que são referidos o horizonte de agendamento, o tamanho do fragmento de agendamento e o tempo de de cada tarefa. Esta propriedade é essencial para que se possa deduzir o baseado na formulação do problema de transportes.

5.3.2 Origens e oferta, destinos e procura

Cada uma das tarefas T_j, a agendar, transformar-se-á numa origem para o problema de transportes. A oferta nessas origens será igual ao número de fragmentos padrão necessários para completar as tarefas. Esse valor pode ser facilmente calculado por:

$$\frac{p'_{sj}}{U'}$$

Sem perda de generalidade, pode admitir-se que o resultado da expressão anterior é um valor inteiro ou, pelo menos, racional.

À semelhança do que se fez para o modelo genérico apresentado, cada processador, durante o horizonte de agendamento, poderá processar, no K fragmentos (processadores mais rápidos). Cada um dos fragmentos, diversos processadores P_i será um destino do problema de transportes. em cada fragmento de cada processador, apenas poderá ser processado fragmento de uma tarefa, a procura em cada um dos destinos será

Para que o modelo fique completo, é necessário evitar que se possa agendar nos últimos fragmentos dos processadores mais lentos, e é necessário fazer com que a soma da oferta nas origens seja igual à soma da procura nos destinos. Estas duas questões são resolvidas da mesma forma, através da adição de uma origem extra. A oferta nessa origem deverá ser igual à diferença entre a soma da procura e a soma da oferta, equilibrando o problema. Esta diferença é dada pela seguinte expressão:

$$m \cdot K - \sum_{j=1}^{n} \frac{p'_{sj}}{U'}$$

Note-se que esta diferença deverá ser sempre igual ou superior a zero. Caso o não seja, é sempre possível aumentar o tamanho do horizonte de agendamento. A oferta nesta origem adicional significa tempo livre a ser atribuído a um ou vários processadores.

Destes fragmentos livres, alguns deverão ser atribuídos aos últimos períodos dos processadores mais lentos, onde não é permitido agendar. A forma de o conseguir passa por colocar na matriz de custos do problema de transportes, nos vectores que representam aqueles períodos, valores arbitrariamente grandes. Apenas se colocará o valor 0 nos elementos desses vectores correspondentes à origem adicional. Desta forma, os últimos períodos dos processadores mais lentos não serão agendados. Sobre a matriz de custos do problema de transportes será feita uma discussão mais aprofundada na secção seguinte.

Exemplo

Considere 3 processadores e 4 tarefas a agendar. Pretende-se fazer o agendamento para 3 períodos padrão e a unidade de agendamento é 1 período padrão, sendo a produção nesse período de 10 unidades padrão. Os coeficientes de velocidade (b_i) para cada processador são 2/3, 1.0 e 4/3. A quantidade a produzir em cada tarefa é de 20 unidades. Os coeficientes de duração da tarefa (c_j) são 0.5, 0.5, 1.0 e 1.5.

O primeiro passo é calcular o número de unidades padrão a produzir (p'_{sj}) para cada uma das 4 tarefas, que é igual ao produto da quantidade a produzir (p'_j) pelo coeficiente de duração da tarefa(c_j). Os resultados serão 10, 10, 20 e 30. Como a produção padrão no fragmento de agendamento considerado é de 10 unidades padrão, a oferta nas 4 origens será 1, 1, 2 e 3. O processador mais rápido é o processador 3, com $b_3 = 4/3$. O valor de K (número de fragmentos padrão que o processador mais rápido pode processar) será igual a 4. A soma da procura será igual a $m \cdot K$, ou seja, será igual a 12.

A oferta na origem adicional será igual à diferença entre a procura (12) e a oferta (7), ou seja será igual a 5.

Recorrendo à apresentação tabular do problema de transportes, ter-se-ia o seguinte:

Origens	Destinos												Oferta
	P_1				P_2				P_3				
	k=1	k=2	k=3	k=4	k=1	k=2	k=3	k=4	k=1	k=2	k=3	k=4	
T_1													1
T_2													1
T_3													2
T_4													3
A													5
Procura	1	1	1	1	1	1	1	1	1	1	1	1	

Nas zonas a sombreado não é possível agendar, pois correspondem aos últimos períodos dos processadores mais lentos. A matriz de custos do problema deverá conter valores arbitrariamente grandes para estes elementos.

Embora a oferta na origem adicional seja igual a 5, o agendamento apenas possui uma folga de 2 fragmentos padrão, uma vez que há 3 fragmentos onde não é possível agendar. ■

A matriz de soluções do problema de transportes possui uma característica importante: é totalmente unimodular[5]. Para esta situação concreta, isto que as variáveis de decisão tomarão um de dois valores: 0 ou 1. Sempre que variável de decisão tome o valor 1, significa que a tarefa corresponde à deve ser agendada no processador e no período correspondente ao destino.

Esta formulação é consistente, uma vez que, num determinado período num determinado processador, apenas pode ser agendado um fragmento de uma tarefa. Por outro lado, a formulação permite a preempção das tarefas e permite o processamento simultâneo, por vários processadores, de uma mesma tarefa.

5.3.3 Matriz de custos

A matriz de custos do problema de transportes é uma peça fundamental da formulação que se apresenta. É na matriz de custos que serão definidas as restrições relacionadas com as datas de disponibilidade e de conclusão e com as adequações entre processadores e tarefas, entre outras.

Designe-se cada variável de decisão do problema de transportes por x_{ijk}, representando, como já se viu, o agendamento ou não de um fragmento da tarefa T_j no processador P_i no período k, conforme a variável assuma o valor de 1 ou de 0, respectivamente. Seja c_{ijk} o elemento da matriz de custos associado à variável x_{ijk}. Note-se que $j = 1,2,\cdots n+1$, devido à origem adicional que se acrescentou ao problema. No entanto, para $j = n+1$, quaisquer que sejam i e k, $c_{ijk} = 0$, fazendo com que esta origem não produza efeitos laterais sobre a solução final.

Na secção anterior já foi sugerido um preenchimento parcial da matriz de custos. Os elementos da matriz correspondentes aos últimos

[5] Ver Nemhauser & Wolsey [15].

períodos dos processadores mais lentos devem ser preenchidos com valores arbitrariamente grandes, excepto os elementos correspondentes à origem adicional, que devem ter custo nulo. Desta forma, seria forçado o não agendamento desses períodos. Matematicamente, para $i = 1,2,\cdots m$, para $j = 1,2,\cdots n$ e para $k = K_i + 1, K_i + 2,\cdots K$, $c_{ijk} = M$, representando M um valor arbitrariamente grande.
A mesma técnica (atribuir custos arbitrariamente grandes) será utilizada para evitar a violação de restrições que não possam ser violadas, tais como incompatibilidades absolutas entre tarefas e processadores, datas de disponibilidade e *deadlines*.

Se uma tarefa, T_j, for incompatível com um determinado subconjunto de processadores I_j, devem ser atribuídos custos arbitrariamente grandes a todos os elementos do vector de custos correspondente a essa tarefa, que digam respeito aos processadores pertencentes ao subconjunto em causa. Ou seja, para $j = 1,2,\cdots n$, para $i : P_i \in I_j$ e para $k = 1,2,\cdots K_i$, $c_{ijk} = M$.

Para introduzir as datas de disponibilidade e as *deadlines*, o procedimento semelhante. No caso das datas de disponibilidade pretende-se que, para determinada tarefa, seja evitado o agendamento no intervalo entre $k = 1$ $k = r_{ij} - 1$. Ou seja, para $j = 1,2,\cdots n$, para $i = 1,2,\cdots m$ e para $c_{ijk} = M$. Relembre-se que r_{ij} é a data de disponibilidade referida em agendamento de um processador específico. No caso das *deadlines*, que seja evitado o agendamento entre $k = \widetilde{d}_{ij} + 1$ e $k = K_i$. Isto é, para $j = 1,2,\cdots n$, para $i = 1,2,\cdots m$ e para $k = \widetilde{d}_{ij} + 1,\cdots K_i$, $c_{ijk} = M$. Mais uma é uma *deadline* referida em períodos de agendamento de um processador específico.

Os períodos de paragem programados também são incluídos no modelo mesma forma. Uma paragem programada pode ser referida como

onde N é o subconjunto de processadores que devem entrar em paragem no período $[k_b, k_e]$. Admita-se que há um conjunto S de paragens programadas. Então, para cada elemento de S, para $j = 1,2,\cdots n$, para $i : P_i \in N$ e para $k = k_{bi}, k_{bi}+1, \cdots, k_{ei}$, $c_{ijk} = M$. Isto equivale a dizer que, nos períodos de paragem programada de um processador, existe um custo arbitrariamente grande para o agendamento de qualquer tarefa. Tal como no modelo genérico, os e de k_{ei} referem-se a períodos de agendamento de um processador

Finalmente, as datas de disponibilidade para os processadores também podem ser modeladas recorrendo a valores arbitrariamente grandes para a matriz de custos. Se a_i for a data de disponibilidade do processador P_i, nada deve ser agendado nesse processador antes do período de agendamento a_i. Isto é, para $i = 1,2,\cdots m$, para $j = 1,2,\cdots n$ e para $k = 1,2,\cdots, a_i - 1$, $c_{ijk} = M$. Pode considerar-se que a_i já está referido em períodos de agendamento de um processador específico.

Note-se que o que foi dito até agora sobre a matriz de custos é uma tradução exacta do que foi dito quando se mencionaram as restrições do modelo genérico, o que ilustra a correspondência que existe entre os dois modelos.

Considerem-se agora os elementos da matriz de custos que não são com o valor arbitrariamente grande, M. Pela análise da estrutura do transportes, é evidente que estes elementos devem ter valores que preferências de agendamento. Por exemplo, considere-se os elementos que dizem respeito a um par processador/tarefa. Considere-se ainda, que existe uma data de conclusão anterior a uma *deadline*. Claramente, é preferível agendar a tarefa antes da data de conclusão, embora seja possível a violação daquela data. Em termos de matriz de custos, os elementos anteriores à data de conclusão devem conter valores mais baixos que os elementos situados depois da data de conclusão. Desta forma estar-se-á a introduzir no problema de transportes a preferência pelo agendamento da tarefa antes da data de conclusão.

Seguindo esta linha de raciocínio, serão introduzidas no modelo as datas de conclusão e as adequações entre tarefas e processadores. Nesta formulação, baseada no problema de transportes, não é possível introduzir, pelo menos de forma óbvia, a questão dos sortidos, a questão dos tempos de preparação e a questão da adjacência física dos teares utilizados.
A introdução de custos pela não adequação entre tarefas e processadores e pela violação das datas de conclusão é feita por adição, ou seja, aos custos que reflectem uma situação são adicionados os custos que reflectem a outra. Inicialmente, todos os elementos da matriz de custos têm o valor zero. Seguidamente, aos elementos afectados pelas restrições descritas anteriormente será atribuído o valor M . Aos restantes elementos serão adicionados os custos relacionados com datas de conclusão e os custos relacionados com inadequações processador/tarefa.

Uma vez que a empresa produz por encomenda, havendo datas de estabelecidas, não há interesse em terminar a tarefa muito mais cedo que data, até porque isso poderia provocar problemas de armazenamento de intermédios. Também não há interesse em terminar a tarefa na data de pois um pequeno imprevisto na produção poderia atrasar o legítimo pensar que, dentro da janela temporal que vai desde a data de disponibilidade até à data de conclusão, é indiferente quando se agenda a tarefa, nada devendo ser adicionado aos respectivos elementos da matriz de custos.

Fora daquela janela temporal e, entre a data de conclusão e a *deadline*, é definir uma função de penalização que forneça os valores a adicionar aos respectivos elementos da matriz de custos. Embora a função a utilizar em última análise, da afinação do modelo, sugere-se uma função exponencial, que consiste em multiplicar uma constante arbitrária pelo períodos de violação. Se W_d for essa constante, para $j = 1,2,\cdots,n$, para para $k = d_{ij}+1, d_{ij}+2,\cdots \widetilde{d}_{ij}$:

$$c_{ijk} = c_{ijk} + \left(k - d_{ij}\right) \cdot W_d$$

Exemplo

Considerem-se os elementos da matriz de custos relativos a determinada tarefa e a determinado processador, compatíveis. A data de disponibilidade da tarefa é 3, a data de conclusão é 6 e a *deadline* é 8 (valores referidos em períodos do processador considerado). Neste caso, para $k = 3$ até $k = 6$, $c_{ijk} = c_{ijk} + 0$. Para $k = 7$, $c_{ijk} = c_{ijk} + W_d$. Para $k = 8$, $c_{ijk} = c_{ijk} + 2 \cdot W_d$.

Repare-se que se a data de conclusão for violada por 1 período o custo associado será igual a W_d. Se for violada por 2 períodos a penalização será, muito provavelmente, igual a $3 \cdot W_d$, devido à estrutura do problema de transportes. Se fosse possível a violação por 3 períodos, a penalidade associada poderia ser $6 \cdot W_d$. Embora os custos colocados na matriz cresçam de forma linear, a penalização, em caso de violação, cresce de forma exponencial. ■

Embora a questão do peso ou importância das várias tarefas não seja ela pode, em parte, ser contemplada nesta fase da formulação. Se houver mais importantes que outras, faz sentido pensar que é preferível violar as datas de conclusão das tarefas menos importantes às das tarefas mais importantes. Logo, em vez de definir uma constante W_d global, poderiam ser definidas constantes W_{dj}, uma por cada tarefa. O valor a atribuir a essas constantes seria proporcional à sua importância relativa. Desta forma estar-se-ia a exprimir a preferência pela violação das tarefas menos importantes.

Como foi afirmado, a escolha da função de penalização deve ser feita em função do comportamento requerido ao modelo, depois de realizar os testes necessários.

A outra componente fundamental da matriz de custos do problema de transportes são as penalidades por inadequação entre processador e tarefa. Relembre-se que, quando foi apresentada a matriz de adequações, designou-por e_{ij} a penalidade por executar a tarefa T_j no processador P_i.

Neste caso a correspondência com a matriz de custos do problema de transportes é óbvia. A forma de reflectir a matriz de adequações na matriz de custos do problema de transportes também é evidente: a todos os elementos da matriz de custos correspondentes à tarefa T_j e ao processador P_i deve ser adicionada a penalidade e_{ij}. Desta forma, sempre que possível, o algoritmo de transportes encontrará os processadores mais adequados para o agendamento de determinada tarefa.
Matematicamente, pode ser escrito: para $i = 1,2,\cdots,m$, para $j = 1,2,\cdots,n$ e para $k = 1,2,\cdots K_i$, $c_{ijk} = c_{ijk} + e_{ij}$.

As constantes e_{ij} também podem ser utilizadas para exprimir o peso ou importância das tarefas. Tal como foi referido para as penalidades violação de datas de conclusão, se as constantes e_{ij} forem mais elevadas tarefas mais importantes, estas tarefas serão preferencialmente agendadas em processadores mais adequados.

5.3.4 Conclusão

Fica assim completamente definida a transformação do problema de agendamento num problema de transportes, com o recurso à relaxação de algumas restrições do problema (tempos de preparação, sortidos, adjacências físicas) e ao arredondamento de datas contínuas para períodos de agendamento discretos.

Quando comparados com a duração das tarefas a executar, tanto os tempos de preparação como os arredondamentos, não têm qualquer significado prático. Por outro lado, desde que se disponha de capacidade computacional, o agendamento pode ser definido com a precisão que se desejar, através da redução do período de agendamento.

5.4 Conclusão

Este capítulo iniciou-se com a apresentação de esboço de um modelo de programação matemática compatível com o problema em análise.

algumas restrições daquele modelo propôs-se um modelo baseado no problema de transportes capaz de encontrar uma solução que, devido às características das restrições relaxadas, é de boa qualidade.
Conforme se demonstrará no próximo capítulo, as soluções conseguidas pelo algoritmo de transportes, sofrem de alguns defeitos que podem ser corrigidos. Estes defeitos surgem devido à relaxação de algumas restrições importantes, como a existência de tempos de preparação na mudança de tarefas.

No próximo capítulo serão apresentadas duas heurísticas que servirão para melhorar as soluções provenientes do algoritmo dos transportes.

Capítulo 6
HEURÍSTICAS DE MELHORIA

Nesta secção serão discutidas duas heurísticas para o melhoramento das soluções fornecidas pelo algoritmo de transportes. Para que se percebam melhor os pontos que irão ser discutidos será introduzida uma pequena instância do problema de agendamento. Este exemplo permitirá também identificar defeitos óbvios na solução introduzida pelo algoritmo de transportes.

6.1 Introdução

A estratégia de desenvolvimento das heurísticas que se apresentam neste capítulo, segue a filosofia exposta no início do capítulo anterior. Ou seja, uma vez encontrada uma formulação capaz de produzir soluções que, pelas suas características, são aceitáveis, é necessário observar atentamente as soluções produzidas e identificar melhorias que se possam fazer. Essas melhorias resultarão da tomada em linha de conta de restrições que não foram consideradas no modelo apresentado no capítulo anterior.

Considere-se então o seguinte exemplo.

Exemplo

Considere-se o problema simplificado, composto por 10 tarefas e 5 processadores, cujos principais dados estão resumidos na tabela que se segue:

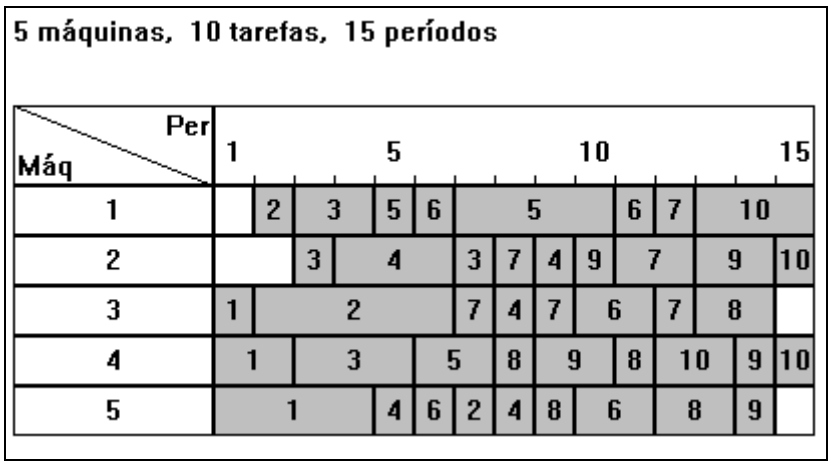
Tarefa (j)	r_j	d_j	p'_{sj}/U'	Adequações				
				P_1	P_2	P_3	P_4	P_5
1	1	6	6	0	0	1	1	1
2	2	7	6	1	0	1	0	1
3	3	8	6	1	1	0	1	0
4	4	9	6	0	1	1	0	1
5	5	10	6	1	1	0	1	0
6	6	11	6	1	0	1	0	1
7	7	12	6	1	1	1	0	0
8	8	13	6	0	0	1	1	1
9	9	14	6	0	1	0	1	1
10	10	15	6	1	1	0	1	0

A tabela inclui, em relação a cada tarefa, as datas de disponibilidade e de conclusão, já expressas em períodos de agendamento padrão, a duração da tarefa, igualmente expressa em períodos padrão, e a matriz de adequações, onde "0" significa "inadequado" ou "agendamento impossível" e "1" significa "perfeitamente adequado". Todos os valores foram escolhidos arbitrariamente.

Considere-se que os processadores funcionam todos à mesma velocidade e que não há restrições adicionais. O horizonte de agendamento é de 15 períodos padrão e a penalidade por violação de datas de conclusão (W_d) é igual a 1.

A solução produzida pela implementação do algoritmo de transportes feita este trabalho[6] foi a seguinte (diagrama de Gantt):

[6] No capítulo seguinte serão introduzidos mais pormenores sobre a implementação computacional do algoritmo de transportes e das heurísticas que agora se descrevem.



Esta é uma solução típica do algoritmo de transportes, da qual se salienta o exagerado número de preempções. O leitor pode facilmente identificar várias preempções claramente desnecessárias. Uma vez que os tempos de preparação não foram incluídos no algoritmo de transportes, a preempção tem naquele algoritmo um custo nulo. A primeira heurística que se irá introduzir visa eliminar um grande número de preempções claramente desnecessárias.■

6.2 Heurística 1

6.2.1 Introdução

Como já se afirmou, a ideia subjacente a esta heurística é a eliminação de preempções desnecessárias. Este objectivo pode ser conseguido através "arrumação" dos vários fragmentos de tarefa agendados em cada considerando cada processador isoladamente. Esta "arrumação" não feita de qualquer maneira: pelo menos, é necessário que as datas de várias tarefas continuem a ser respeitadas ou, no caso de já existirem que estas não aumentem. Ou seja, é necessário encontrar uma solução mesmos atrasos da solução original, uma vez que é impossível encontrar solução com menos atrasos (dizer que é possível encontrar uma solução menos atrasos é equivalente a dizer que o algoritmo de transportes não

solução óptima, uma vez que os atrasos são fortemente penalizados algoritmo).

Pretende-se então, encontrar uma solução com os mesmos atrasos e, se possível, com menos preempções. Demonstrar-se-á nos parágrafos seguintes que optimizar o número de preempções e, ao mesmo tempo, não aumentar os atrasos, é um problema NP-completo. Entre fazer uma preempção desnecessária (incorrer num tempo de preparação) e atrasar uma tarefa, é claramente preferível fazer a preempção. O algoritmo a utilizar deverá, impreterivelmente, manter os atrasos da solução e, como segundo objectivo, minimizar o número de preempções.

6.2.2 Apresentação da heurística 1

Atente-se agora no exemplo apresentado. Considere-se a máquina 4. A sequência de tarefas agendada é 1-1-3-3-3-5-5-8-9-9-8-10-10-9-10. A sequência poderia ser, sem qualquer violação de datas de disponibilidade ou de conclusão, 1-1-3-3-3-5-5-8-8-9-9-9-10-10-10. Comparando, as tarefas 1, 3 e 5 não são alteradas, a tarefa 8 começa no mesmo período mas acaba mais cedo, a tarefa 9 começa mais tarde e acaba mais cedo e a tarefa 10 começa mais tarde, acabando no mesmo período. A sequência original exige que o processador seja preparado 8 vezes (admitindo que inicialmente está preparado para a tarefa 1), enquanto que na segunda sequência apenas são necessárias 5 preparações adicionais. Nos outros processadores poderiam ser feitas economias semelhantes.

Mas atente-se agora no processador 3, em que a sequência original é 1-2-7-4-7-6-6-7-8-8. Note-se que a tarefa 8 está a violar a data de conclusão período. Fazendo a agregação das tarefas a começar no início, ficaria 1-7-7-7-4-6-6-8-8. As tarefas 4 e 6 passaram a violar as datas de conclusão. tarefas forem agregadas começando no fim, a sequência ficará 1-2-2-2-2-

7-7-7-8-8, não havendo assim violações adicionais. Pretende-se agregação das tarefas não pode ser feita de forma arbitrária, sob o risco incorrer em violações de datas de disponibilidade ou de conclusão.

Não é difícil imaginar uma situação em que, quer se agregue num sentido ou no outro se incorrerá em violação (infelizmente, o exemplo apresentado não possui uma situação desse tipo). Considere-se a sequência 1-1-2-2-1-1, em que a data de disponibilidade da tarefa 2 é igual a 3 e a data de conclusão é igual a 4. Neste caso, a tarefa 2 nem pode ser executada antes da tarefa 1 nem depois da tarefa 1, sem se incorrer em violações, ou seja, é necessário manter as preempções que existem tal como estão.

Fundamentalmente, o que se pretende fazer é resolver, para cada processador o problema $1\left|pmtn, r_j\right|\sum D_j$, ou seja, pretende-se resolver um problema com um processador, com tarefas preemptivas sujeitas a datas de disponibilidade, para a minimização do total de atrasos (*tardiness*). Adicionalmente, pretende-se minimizar o número de preempções. Este problema é equivalente ao Problema do Caixeiro Viajante (PCV) com janelas temporais.

Considere-se que cada fragmento das várias tarefas é um vértice do Considere-se ainda um vértice adicional que será o ponto de partida e o chegada. Todos os arcos ligados ao vértice adicional tem um custo nulo. que liguem dois vértices correspondentes a fragmentos de uma mesma também devem ter custo nulo. Os restantes arcos, que ligam vértices correspondentes a fragmentos de tarefas distintas, devem ter um custo janela temporal de cada vértice corresponderá ao intervalo entre a data disponibilidade e a data de conclusão da tarefa originária do respectivo fragmento. Se a data de conclusão estivesse em violação na solução deveria ser substituída pela data efectiva de conclusão, tornando desta

solução possível. A minimização da "distância" percorrida corresponde minimização do número de preempções a efectuar. Como é possível todas as janelas temporais (a solução original respeita-as), os atrasos superiores aos atrasos da solução original.

Como o PCV (sem janelas temporais) é NP-completo[7], o PCV com temporais tem, pelo menos, o mesmo grau de complexidade, pois é uma generalização do PCV sem janelas temporais. Devido a esta PCV, a solução óptima seria dispendiosa a nível computacional. O encontrar um algoritmo que, sem aumentar os atrasos, diminua o preempções sem, necessariamente, o optimizar.

Considere-se novamente o problema $1|pmtn, r_j|\sum D_j$. Se fornecermos a um algoritmo de optimização datas de conclusão modificadas para as tarefas atrasadas, ou seja, em vez de fornecer a data real de conclusão, fornecer a data de conclusão mais o atraso (tal como na transformação para PCV), verifica-se que, na solução óptima $\sum D_j = 0$. Isto implica que $L_{\max} \leq 0$. Então, resolvendo o problema $1|pmtn, r_j|L_{\max}$, obter-se-á uma solução em que $\sum D_j = 0$, que é precisamente o objectivo.

Considere-se o caso do agendamento de uma máquina, com o objectivo minimizar o atraso máximo ($L_{\max}$), em que não há datas de em que as tarefas a agendar vão sendo conhecidas à medida que o ou seja, de forma dinâmica. A solução óptima para este problema pode através de uma regra de despacho, conhecida por regra PDEDD[8] *dynamic earliest due date*).

[7] Ver Garey e Johnson [7]
[8] Ver Morton e Pentico [14]

A regra PDEDD consiste em agendar, de entre as tarefas disponíveis em determinado instante, aquela que tiver menor data de conclusão. A qualquer momento, se ficar disponível uma tarefa com menor data de conclusão que a tarefa em curso, a tarefa em curso deve ser interrompida e deve ser agendada a tarefa entretanto disponível.

Esta regra também produz soluções óptimas no caso não dinâmico, para o problema $1|pmtn, r_j|L_{\max}$ (ver Blazewicz e al. [3]). Neste problema, as tarefas ficam disponíveis para agendamento na respectiva data de disponibilidade.

Transpondo a regra PDEDD para o ambiente do presente trabalho, obtém-se: percorrendo o horizonte de agendamento período a período, do início para o final, agendar em cada período um fragmento da tarefa que, entre as tarefas disponíveis nesse período e não completamente agendadas, tiver menor data de conclusão. Se não houver nenhuma tarefa disponível, não agendar nada no período.

A sequência produzida através da regra PDEDD é óptima em termos de $L_{\max}$, e portanto, como se demonstrou, nunca apresentará violações de datas de conclusão superiores (nem inferiores) às violações da sequência original.

Porém, é legítimo questionar se aquelas violações ocorrem nas tarefas ocorriam na solução original. De facto, é importante que isto aconteça, importante que a violação ocorra na mesma tarefa. Imagine-se a situação determinada tarefa tem a data de conclusão violada em dois período. Depois de aplicar a regra PDEDD, num dos processadores, a de 1 período passou para uma outra tarefa. Há agora uma situação de 1 período em duas tarefas distintas. A primeira situação parece medida em que, globalmente, há apenas uma tarefa atrasada 1 período

processadores diferentes). Na segunda situação há duas tarefas atrasadas período cada.

A forma de "obrigar" a regra PDEDD a violar as datas de conclusão das tarefas originalmente violadas é conseguida quando, em vez de fornecer à regra PDEDD as verdadeiras datas de conclusão das tarefas, para as tarefas que estão em violação, na máquina em análise, se fornece uma data de conclusão igual à data de conclusão verdadeira mais o número de períodos de violação, ou seja fornecer à regra PDEDD um $d'_j = max(d_j, C_{ij})$, em que C_{ij} é o instante de conclusão da tarefa j no processador i. Este ponto será objecto de uma análise mais detalhada na próxima secção.

A implementação que foi feita da regra PDEDD, aplicada ao exemplo apresentado no início do capítulo e à solução original, também apresentada, produz o seguinte resultado:

5 máquinas, 10 tarefas, 15 períodos

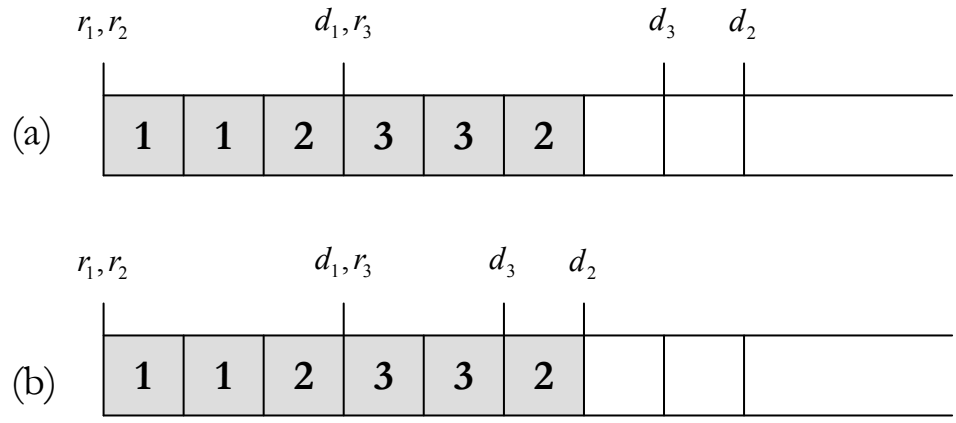
Máq \ Per	1	5	10	15
1	2	3	5	6 7 10
2	3	4	7	9 10
3	1	2	4 6 7	8
4	1	3	5 8 9	10
5	1	2	4 6	8 9

Pode ser observada uma diminuição significativa do número de preempções em relação à solução original, o que faz supor que a regra PDEDD produz bom resultados. Esta afirmação será confirmada pelos testes computacionais realizados, que serão apresentados nos capítulos seguintes.

6.2.3 Considerações finais à heurística 1

A presente heurística consiste na aplicação da regra PDEDD aos resultados produzidos pelo algoritmo de transportes. Indirectamente, espera-se que a heurística reduza o número de preempções desnecessárias. Se tal não acontecer, e o número de preempções aumentar, pode-se sempre repor a solução original.

A solução produzida por esta heurística pode ainda conter preempções desnecessárias. No entanto, estas serão, em princípio, em número significativamente menor que na solução original. De facto, estas preempções eventualmente desnecessárias só acontecem se, para duas quaisquer tarefas, $r_1 < r_2$ e se $d_1 > d_2$. Neste caso, a preempção acontecerá, mas nem sempre será desnecessária, dependendo das condições do problema concreto. Para ilustrar a situação apresentam-se dois exemplos:



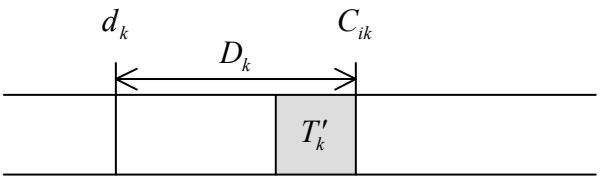
No exemplo (a), a preempção da tarefa 2 é desnecessária, havendo outras soluções de agendamento sem aquela preempção e sem atrasos. Por exemplo, poder-se-ia deixar o terceiro período de agendamento livre e agendar a tarefa 2 depois da tarefa 3. Também se poderia agendar a totalidade da tarefa 2 antes de iniciar a tarefa 3. Já no exemplo (b), a preempção não pode ser evitada, sob pena de um aumento nos atrasos nas tarefas.

Relembre-se que, no ambiente da fábrica analisada, as datas de disponibilidade, no agendamento de curto prazo, apenas surgem em situações excepcionais, como por exemplo um erro na gestão de existências ou um atraso significativo de um fornecedor. No caso limite de não haver datas de disponibilidade, o número de preempções também é optimizado.

Tal como já foi afirmado, os testes computacionais realizados confirmam que a heurística 1 tem um efeito significativo na redução de preempções desnecessárias, embora, tal como ficou demonstrado, não seja garantidamente óptima.

Um ponto que não foi completamente dissecado quando da apresentação da heurística, na secção anterior, tem a ver com as tarefas que são violadas na nova solução. De facto, se fornecermos à heurística uma data de conclusão modificada, $d'_j = max(d_j, C_{ij})$, as violações ocorrerão, necessariamente, nas mesmas tarefas.

A afirmação do parágrafo anterior pode ser demonstrada com um argumento simples. Considere-se a solução proveniente do algoritmo de transportes, que é óptima no critério $\sum D_j$. Embora possa haver várias soluções óptimas, considere-se aquela que o algoritmo fornece. Considere-se ainda um determinado processador em que um fragmento de tarefa T'_k está atrasado $D_k > 0$.



A única forma de diminuir o atraso do fragmento T'_k é agendá-lo na (antes do momento em que está agendado) do agendamento. Na parte agendamento não pode existir nenhum fragmento de tarefa T'_j tal que pois, caso existisse, o algoritmo de transportes teria antecipado T'_k e

no lugar de T'_k, diminuindo a função objectivo gratuitamente. Logo, a forma de reduzir o atraso de T'_k é por troca com um fragmento de tarefa que $d_j < C_{ik}$. Como se fornecem à heurística datas de conclusão fornecer-se-iam $d'_j = d_j$ e $d'_k = C_{ik}$. Como $C_{ik} > d_j$, pela regra fragmento T'_k nunca seria agendado antes de qualquer fragmento T'_j na parte inicial da solução. Também nunca seria agendado depois do em que está agendado pois isso resultaria em $L_{\max} > 0$. Pode então que os atrasos da solução original são mantidos, exactamente nas e na mesmas grandezas, na solução pós heurística.

Em vez de fornecer à regra PDEDD datas de conclusão modificadas, tais que, $d'_j = max(d_j, C_{ij})$, há outra alternativa. Ao entrar com C_{ij} no cálculo das datas modificadas, ignora-se que as tarefas possam ter as datas de conclusão violadas em outras máquinas. Por outro lado, e embora possa ser discutível, não parece muito grave violar uma data de conclusão que já está a ser violada noutros processadores. O que aconteceria ao fazê-lo era tornar o atraso mais difícil de recuperar. Por estes motivos, na versão da regra PDEDD que foi implementada, utilizaram-se datas de conclusão modificadas calculadas como $d'_j = max(d_j, C_j)$ em que C_j é o instante de conclusão da tarefa no(s) processador(es) que a terminam mais tarde.

Neste caso, já não é garantido que os atrasos ocorram exactamente nas tarefas e nas mesmas grandezas. O que pode acontecer é a transferência atrasos, dentro de um mesmo processador, entre tarefas atrasadas. argumento utilizado para demonstrar a primeira situação, agora já pode fragmento de tarefa T'_j, tal que $d_j \geq C_{ik}$. Só que esse fragmento não pertencer a uma tarefa não atrasada, pois, caso pertencesse, o algoritmo

transportes teria antecipado o agendamento do fragmento T'_k por troca mais uma vez, diminuindo a função objectivo gratuitamente. Logo, só trocas de atrasos entre duas tarefas atrasadas. Conforme se verificará forem apresentados os testes computacionais, esta propriedade produz sobre a solução que se podem considerar positivos.

Uma outra questão prende-se com a utilização de pesos para as tarefas. Se for utilizada esta última versão do cálculo de datas de conclusão modificadas, pode haver transferência de atrasos de tarefas menos importantes para tarefas mais importantes. Logo, caso esta questão seja muito importante, é de considerar a primeira versão do cálculo, que resulta na impossibilidade de haver transferências de atrasos entre tarefas.

6.2.4 Pseudocódigo

Apresenta-se agora o pseudocódigo da heurística 1, tal como foi proposta nas secções anteriores. Relembre-se que a heurística parte de uma solução proveniente do algoritmo dos transportes, que consiste num plano de agendamento para m máquinas. Cada processador P_i possui K períodos de agendamento onde estão agendados fragmentos de tarefas. A heurística parte de uma lista T' composta por todos os fragmentos agendados, na solução de partida, numa determinada máquina. A determinação da solução é feita período a período, sendo a todo o momento mantida uma lista, E, de fragmentos que podem ser agendados num determinado período. A esta lista E vão sendo adicionados os fragmentos de tarefa que vão ficando disponíveis e vão sendo retirados os fragmentos de tarefa que vão sendo agendados na solução final.

```
begin
for i:=1 to m do
 begin
 T':={fragmentos agendados em P_i}
```

```
for j = 1 to n do
 begin
   d'_j := max(d_j, C_j);
 end;
E := ∅;
for k := 1 to K do
 begin
 E := E ∪ {T'_j : r_j = k};
 if  E = ∅
 then não agendar nada no período k;
 else
  begin
   escolher T'_x ∈ E  tal que  d'_x = min_{T'_j ∈ E} {d'_j};
   agendar T'_x no período k;
   E := E − {T'_x};
  end;
 end;
end;
end;
```

A complexidade desta heurística é de $O(mK^2)$. Há um ciclo geral que é executado uma vez por máquina, logo, de complexidade $O(m)$. Dentro desse ciclo, é feito o cálculo dos d'_j, havendo, no máximo, K fragmentos e sendo a complexidade da operação $O(K)$, e é executado um ciclo onde, para cada período de agendamento, é escolhido um fragmento, o que tem uma complexidade de $O(K^2)$. Logo, a complexidade da heurística é de $O(mK^2)$, o que é muito eficiente, mesmo para valores elevados de m ou de K. Refira-se que, se a heurística for executada para uma única máquina, tem uma complexidade de $O(K^2)$.

6.3 Heurística 2

6.3.1 Introdução

O objectivo desta segunda heurística é fazer a melhoria das soluções algoritmo de transportes e melhoradas pela primeira heurística,

secção anterior. Assim, numa primeira fase é necessário identificar os defeitos que se pretendem eliminar da solução. Tal como na heurística 1, pretende-se eliminar tempos de preparação desnecessários, introduzidos na solução pelo algoritmo de transportes que, como já se verificou, é insensível ao número de preparações realizadas.
Enquanto que a heurística 1 se focou na eliminação de tempos de preparação desnecessários partindo do conjunto de tarefas agendadas num único processador, esta segunda heurística tentará eliminar preparações efectuando trocas de tarefas entre processadores diferentes.

Verificou-se na secção anterior que a minimização do número de preparações numa única máquina era um problema NP-completo. Como o problema de uma única máquina é um caso particular do problema de múltiplas máquinas, pode admitir-se que a minimização do número global de preparações da solução é um problema, pelo menos, tão difícil. Devido à complexidade do problema, propor-se-á então uma heurística capaz de diminuir o número de preparações, mas sem a ideia de optimização.

6.3.2 Apresentação da heurística 2

Para ilustrar o tipo de defeitos que a solução produzida pelo algoritmo transportes e melhorada pela heurística 1 ainda possui, atente-se na a aplicação da heurística 1. Na máquina 3 está agendado um fragmento 1) da tarefa 4. Se, por hipótese, se conseguisse agendar esse fragmento numa outra máquina, onde já estivesse agendada a tarefa 4, por troca fragmento de uma tarefa já agendada na máquina 3, existiria uma economia de um tempo de preparação na máquina 3, ou seja, não seria necessário preparar a máquina 3 para a tarefa 4. Na outra máquina, não haveria nenhum acréscimo de tempos de preparação, uma vez que a tarefa 4 já lá estava agendada.

Partindo do facto de a solução poder conter, agendados em diversos processadores, pequenos fragmentos de tarefas, tentar-se-á definir um rápido, para efectuar uma busca sistemática de pequenos fragmentos de

(que podem ser de tamanho superior a um período de agendamento) e para procurar máquinas e tarefas que possam ser objecto de troca. O algoritmo efectuaria as trocas viáveis de modo a melhorar a solução por diminuição do número de preparações.
Uma troca de fragmentos de tarefa entre processadores torna-se inviável se houver aumento de atrasos em alguma das máquinas em causa na troca. Ainda no exemplo que vem a ser seguido, na máquina 1 existe um fragmento da tarefa 2. Este fragmento nunca pode ser movido, por troca, para outra máquina, pois nenhuma das tarefas agendadas na máquina 1 (excepto a tarefa 2) está disponível no segundo período. No limite, agendar mais um fragmento de qualquer das tarefas agendadas, iria atrasar a tarefa 10 por 1 período. Como é evidente, a eliminação de fragmentos só deve ser realizada, se a solução resultante for claramente melhor, o que não é o caso. Aqui, a expressão "claramente melhor" pretende significar uma solução com menos preparações mas com os mesmos atrasos da original.

Para que se possa eliminar o fragmento da tarefa T_x do processador P_x é necessário reunir algumas condições:

– Encontrar um processador P_y onde esteja agendada a tarefa T_x;

– Encontrar uma tarefa T_y que esteja agendada no processador P_x e no processador P_y, ou encontrar um tempo livre em P_y;

– Verificar se a troca não resulta em violações nas datas de conclusão de qualquer das tarefas agendadas, quer no processador P_x, quer no processador P_y.

As duas primeiras condições podem ser preenchidas através de um simples de busca sequencial. Sempre que se encontrar um par de que satisfaça as duas primeiras condições, deve verificar-se se a terceira é satisfeita.

Esta terceira verificação é bastante simples: basta aplicar a regra PDEDD à nova carga de tarefas de ambas as máquinas, fornecendo datas de conclusão de acordo com a regra definida na secção anterior. Como a regra PDEDD minimiza o atraso máximo, $L_{\max}$, só não há violação de datas de conclusão superiores às da solução original se $L_{\max}$, depois de efectivada a troca continuar igual a zero. Logo, a solução após a troca só é claramente melhor que a solução anterior se, da aplicação da regra PDEDD aos dois processadores envolvidos, resultarem valores de $L_{\max}$ iguais a zero. Apenas nesta situação se deve efectivar a troca.
Uma vez que a eliminação de uns fragmentos pode inviabilizar a eliminação de outros, faz sentido inquirir sobre quais os fragmentos a eliminar em primeiro lugar. Faz também sentido perguntar qual deve ser a troca a efectivar, caso haja várias alternativas. A resposta cabal a estas questões poderia ser dada através de uma pesquisa em árvore ou de um qualquer método de enumeração. Seria uma resposta dispendiosa em termos de recursos computacionais. Por outro lado, não há a certeza da mais valia que um procedimento desse tipo iria significar.

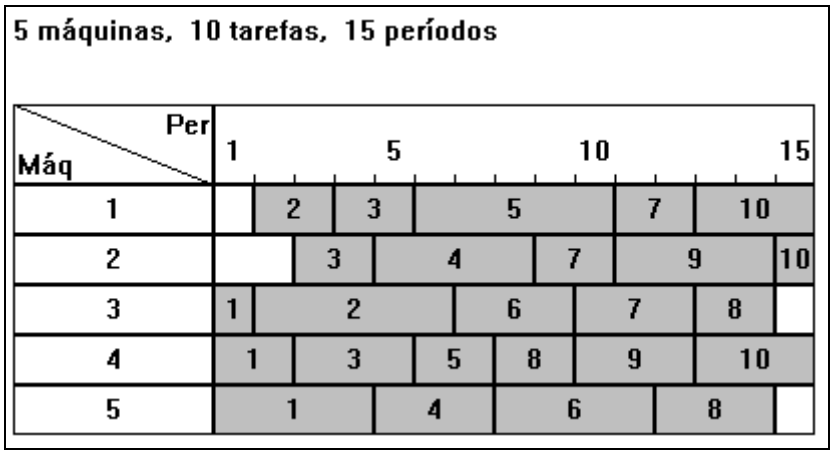
Logo, como o preenchimento das duas condições iniciais para a efectivação de uma troca exige um procedimento de pesquisa pela solução, o que se propõe é a efectivação da primeira troca que satisfaça todas as condições, num procedimento claramente "ganancioso".
Surge ainda a questão sobre qual deverá ser o tamanho máximo do fragmento a eliminar. Esta questão deve ser analisada à luz da relação entre os tempos médios de preparação e o tamanho do período de agendamento. Se o fragmento de agendamento for muito pequeno, o fragmento a eliminar deve ser composto por um conjunto razoável de períodos de agendamento, de modo a que não se incorra em tempos de preparação para durações muito pequenas. Se o período de agendamento for muito grande, a própria aplicação da heurística pode ser irrelevante, pois o tempo de preparação é insignificante em relação ao tamanho do período de agendamento.

Uma vantagem desta heurística é a impossibilidade de entrar em ciclo. sempre que se efectiva uma troca, há uma máquina que fica com um

tarefas agendadas mais restrito. Esta propriedade permite aplicar a heurística de forma recorrente, para eliminar fragmentos de várias dimensões. Por exemplo, admita que se deseja eliminar fragmentos com tamanho até dois períodos de agendamento. Poder-se-ia começar por eliminar todos os fragmentos de tamanho 1 e passar depois aos fragmentos de tamanho 2. Como a eliminação de fragmentos de tamanho 2 pode conduzir à criação de novos fragmentos de tamanho 1, poder-se-ia correr a heurística várias vezes, alternando entre fragmentos de tamanho 1 e de tamanho 2, até que não fossem efectivadas trocas. Outra alternativa seria limitar o número de passagens da heurística, caso verificasse que a heurística era um processo dispendioso a nível

Outra questão prende-se com a existência de máquinas mais adequadas e de máquinas menos adequadas para a realização das tarefas. Assim, ao efectuar as trocas, as tarefas podem passar a ser agendadas em processadores menos adequados, acarretando desta forma uma menos valia para a solução. No caso da implementação realizada, este aspecto não foi tomado em consideração, por não se considerar importante. No entanto, em caso diverso, se esta questão for muito importante e se desejar evitar trocas para máquinas menos adequadas, basta introduzir no algoritmo uma verificação adicional, de modo a comparar a adequação das máquinas envolvidas. Se uma troca necessitar de agendar em máquinas menos adequadas, há sempre a opção de não efectivar a

Referindo o exemplo que tem vindo a ser tratado, a aplicação da heurística 2 de modo a tentar eliminar os fragmentos de tamanho igual a 1 período de agendamento, resultou na seguinte solução:



Compare-se esta solução com a solução que se obteve após aplicar a heurística 1 e verifique-se a eliminação de alguns fragmentos de tamanho igual a 1 período de agendamento, produzindo uma solução igual no fundamental, mas com menos tempos de preparação. Conforme se irá verificar ao analisar os testes computacionais realizados, a redução do número de preparações pode ser considerada significativa

6.3.3 Pseudocódigo

Como a heurística 2 é algo complexa, sob o ponto de vista computacional, havendo várias possibilidades de implementação, em vez de apresentar um pseudocódigo detalhado, apresentam-se apenas, e de forma mais descritiva, os passos a seguir numa implementação.

Passo 1: Ordenem-se os processadores por uma qualquer ordem arbitrária e denominem-se por P_1, P_2, ... P_n. Faça-se $P_I = P_1$.

Passo 2: Começando no processador P_I, percorrer a solução processador a processador e período a período até encontrar um fragmento de tarefa de tamanho alvo. Denote-se a respectiva tarefa por T_x e o respectivo processador por P_x. Se não for encontrado um fragmento de tamanho alvo, terminar.

Passo 3: Começando no processador P_1 e excluindo o processador P_x, percorrer a solução, processador a processador e período a período até encontrar um processador, P_y, onde esteja agendada a tarefa T_x. Se não for encontrado, retomar o passo 2.

Passo 4: Verificar se existe uma tarefa T_y (ainda não testada), agendada quer no processador P_x, quer no processador P_y. Se não existir, ir para o passo 7.

Passo 5: Verificar se é possível efectuar a troca do fragmento T_x por um fragmento de T_y de tamanho igual, agendando o fragmento T_x em P_y e agendando o fragmento necessário de T_y em P_x. Se não for possível regressar o passo 4.

Passo 6: Verificar, tanto em P_x como em P_y, se os atrasos não aumentaram. Se não aumentarem, consumar a troca e ir para o passo 2 com $P_I = min(P_x, P_y)$. Se aumentarem regressar ao passo 4.

Passo 7: Verificar se existe em P_y tempo livre suficiente para agendar T_x. Se não existir retomar o passo 2.

Passo 8: Agendar T_x em P_y e verificar se os atrasos aumentam. Se não aumentarem, consumar a troca. Em todo o caso, retomar o passo 2.

A complexidade deste heurística também pode ser avaliada. O passo 1 qualquer operação, pois os processadores, normalmente, são passo 2 é feita a pesquisa dos fragmentos, que poderão ser, no máximo Logo é uma tarefa de complexidade $O(mK)$. O passo 3 também é uma complexidade $O(mK)$, pois trata-se de encontrar outros fragmentos na No passo 4 é feita uma tentativa para encontrar um fragmento em duas diferentes, que tem uma complexidade de $O(2K)$. Os passos 5 e 6 tem complexidade de $O(K^2)$, pois, fundamentalmente, trata-se de executar a PDEDD para as máquinas envolvidas. O passo 7 é de complexidade é necessário percorrer a solução de um processador. Finalmente, o passo novamente uma verificação utilizando a regra PDEDD (complexidade Os passos 7 e 8 são executados em alternativa aos passos 6 e 7, pelo que complexidade do conjunto de passos 5, 6, 7 e 8 é de $O(K^2)$. A global da heurística é de $O(m^2K^5)$, que é um valor elevado.

Possivelmente, recorrendo a estruturas de dados mais elaboradas, a complexidade desta heurística poderia ser reduzida. Estas possibilidades não foram consideradas porque o tempo de execução da heurística era razoavelmente pequeno e perfeitamente aceitável.
Refira-se ainda que a complexidade estudada diz respeito ao pior caso. Em termos médios, a heurística é menos complexa, uma vez que há a possibilidade de efectuar trocas sem ter de percorrer toda a solução.

6.4 Conclusão

Este capítulo conclui a apresentação dos algoritmos para obtenção de uma solução para o problema de planeamento operacional em estudo.

Globalmente, embora o conjunto de algoritmos propostos não leve à solução óptima, até porque algumas condicionantes, como a questão dos sortidos e das distâncias físicas entre teares, não foram consideradas, estão reunidas as condições para obtenção de soluções de boa qualidade. Neste caso, a qualidade da solução, e de eventuais melhorias que se possam fazer no futuro, apenas pode ser avaliada pela empresa, através da comparação com o passado.

No próximos capítulos, será apresentada a implementação computacional fez dos algoritmos propostos e serão analisados os resultados de alguns foram realizados com o auxílio dessa mesma implementação.

Capítulo 7
IMPLEMENTAÇÃO COMPUTACIONAL

A implementação computacional dos algoritmos descritos nos capítulos anteriores consubstanciou uma parte significativa do trabalho realizado. Este capítulo é dedicado à descrição da implementação efectuada.

Ao longo do capítulo serão explicadas as principais opções que foram feitas ao nível da programação e serão explicadas, de forma sucinta, a estrutura do programa e os seus princípios de funcionamento.

7.1 Linguagem de programação

Para fazer a implementação computacional dos modelos optou-se por uma linguagem de programação de alto nível. De entre as várias possibilidades foi escolhido o *Microsoft Visual C++*, versão 6.0.

Entre várias vantagens que esta linguagem apresenta, podem citar-se algumas:

– Facilidade de utilização de objectos visuais, que conferem aos programas um aspecto "estilo *Windows*", facilitando a interacção com o utilizador;

– Possibilidade de utilização como uma linguagem de programação normal (variáveis, estruturas de controlo, etc.);

– Possibilidade de implementar rotinas de baixo nível para aumentar a eficiência (ponteiros, alocação dinâmica de memória, etc.).

Todas estas vantagens, aliadas à facilidade de programação (de facto, esta linguagem é tão fácil de programar como o Pascal ou o Fortran, só para linguagens conhecidas) tornaram esta linguagem uma excelente escolha para a implementação computacional dos modelos.

7.2 Descrição da implementação

A implementação foi baseada em objectos de programação. Grosso modo, um objecto é composto por uma estrutura de dados e por rotinas que manipulam esses dados. Nesta linguagem de programação, os objectos também são denominados por "classes".

Na figura podem ver-se as classes que foram utilizadas para fazer a implementação computacional. As classes *CAboutDlg*, *CDissertApp* e

implementam funcionalidades relacionadas com o sistema operativo e foram geradas automaticamente. A classe *CDissertDoc* implementa as funcionalidades relacionadas com a manipulação de documentos (neste caso, as instâncias), tais como o criar, o abrir e o salvar. A classe *CDissertView* implementa o interface gráfico do programa. As várias classes *Dialogo** implementam as várias caixas de diálogo apresentadas pela programa ao utilizador. Finalmente, as classes *CInstancia* e *CRelax4* implementam os algoritmos básicos do programa e são o núcleo do próprio programa, daí merecerem uma descrição mais detalhada.
Na classe *CRelax4* é implementado o algoritmo dos transportes. Para essa implementação foi utilizado um código de domínio público denominado RELAX IV. O código serve para resolver problemas de fluxo de custo mínimo e a sua descrição encontra-se em Bertsekas [2].

Embora o código esteja publicado em Fortran, como se sabe, os programas Fortran são pré-compilados para C. O que se fez foi a pré-compilação do para C, seguida de pequenas alterações devido a questões de alterações nas rotinas de entrada e saída de dados. A entrada e saída de era feita através de ficheiros de texto, passou a ser assegurada por rotinas escritas para o efeito que transferem os dados de entrada, da estrutura de dados principal para o código RELAX IV e transferem os dados de saída, novamente para a estrutura de dados principal. Estas rotinas são parte integrante da classe *CInstancia*. De resto, o código RELAX IV foi tratado como uma caixa negra, ao qual se fornecia os dados de entrada e se extraia os dados de saída.

O motivo principal que levou à escolha deste código prende-se com a grande eficiência que o código demonstra em instâncias do problema de transportes de grande dimensão. A adicionar a isto pode referir-se a relativa facilidade com que o código foi inserido no programa desenvolvido.

A classe *CInstancia* é nuclear. É nesta classe que está definida a estrutura de de uma instância e estão definidas todas as rotinas que manipulam esses dados. Nesta classe estão definidas rotinas que vão desde simples procedimentos para alterar dados relativos a uma tarefa ou a uma máquina,

rotinas que implementam os procedimentos de ligação ao módulo *CRelax4* e às rotinas que implementam as heurísticas 1 e 2. Na figura ao lado é possível ver mais algumas rotinas da classe *CInstancia*, cujos nomes são mais ou menos elucidativos do propósito. Pode dizer-se que é nesta classe que se encontra o "cérebro" da aplicação.

7.2.1 Principais rotinas

Embora a aplicação desenvolvida compreenda um vasto conjunto de efectuam as mais variadas tarefas de manipulação de dados, as rotinas importantes são aquelas que implementam os algoritmos apresentados capítulos anteriores. Estas rotinas são basicamente três: a rotina *Relax* implementa o algoritmo dos transportes, e as rotinas *Heuristica1* e implementam as heurísticas apresentadas no capítulo anterior.

Quanto à rotina *Relax*, esta limita-se a recorrer à classe *CRelax4* para fazer executar o já referido código de domínio público RELAX IV. Este rotina tem duas importantes rotinas de suporte, a rotina *WriteToRelax*, que transfere os dados do programa principal para a classe *CRelax4*, e a rotina *ReadFromRelax*, que faz o oposto, ou seja, transfere as soluções do problema de transportes para o programa principal. Confirmou-se a grande rapidez do código na resolução de problemas de transportes de grande dimensão.
A rotina *Heuristica1* limita-se a executar a rotina *RegraPDEDD* para todas as máquinas do plano de agendamento. É na rotina *RegraPDEDD* que está implementada a regra PDEDD que foi apresentada no capítulo anterior. Esta implementação tem um tempo de execução muito rápido por se tratar de uma manipulação de dados bastante simples.

A rotina *Heuristica2* implementa o conjunto de procedimentos que foram definidos no capítulo anterior como Heurística 2. Sempre que é troca potencial, passível de eliminar um fragmento de um processador, é a rotina *TrocarTarefas* que, por sua vez, recorre à rotina *RegraPDEDD* verificação da possibilidade da troca. A rotina *Heuristica2* tem um tempo execução relativamente grande e algo variável de instância para instância,

entre instâncias com características idênticas, o que faz supor que a implementação feita ainda terá alguma margem para melhoria.

Embora a implementação computacional efectuada não tenha o nível de implementação profissional (por exemplo, a rotina que implementa a parece algo demorada, em termos de tempo de execução), revelou-se ferramenta preciosa na afinação dos modelos desenvolvidos. O interface que foi implementado (que será apresentado na próxima secção) economias de tempo, em comparação com, por exemplo, uma entrada e saída de dados através de um ficheiro de texto.

Na próxima secção é feita uma descrição dos aspectos mais relevantes da interface gráfica e são mostradas as principais funcionalidades da implementação computacional desenvolvida.

7.3 Interface

Embora o interface desenvolvido tenha um interesse meramente académico, uma vez que a implementação da aplicação em ambiente industrial necessitaria da implementação de comunicações com o sistema de informação da empresa, bem como do desenvolvimento de um interface de utilização adequado aos operadores da empresa, ele foi extremamente útil no desenvolvimento dos modelos.

Como se pode ver na figura ao lado, a aplicação tem o aspecto (e também as funcionalidades) característico das aplicações do *Microsoft Windows*.

A aplicação possui seis menus, sendo quatro deles típicos de qualquer
O menu *File* serve para criar, abrir e guardar documentos (instâncias). No *Edit* não foi activada qualquer função. No menu *View* podem ser expressas

algumas preferências quanto à apresentação dos resultados. O menu *Help* dá acesso à ajuda de utilização da aplicação.

Para criar uma nova instância é necessário introduzir todos os dados relevantes dessa mesma instância. Essa tarefa deve ser feita recorrendo às caixas de diálogo do menu *Instância* (figura seguinte).

Para definir completamente uma instância, inicialmente, devem ser o número de processadores, o número de tarefas e o número de períodos agendamento a programar. O período de agendamento pode ser uma qualquer unidade de tempo. Estes valores são especificados na caixa de diálogo *Parâmetros...*.

De seguida devem ser introduzidos os dados relativos às tarefas (duração, datas de disponibilidade e de conclusão) na caixa de diálogo *Tarefas...*, e os dados relativos aos processadores (indisponibilidades e tarefas em curso) na caixa de diálogo *Máquinas...*.

Finalmente, deve ser activada a caixa de diálogo *Adequações...* para introduzir a matriz de adequações. Nesta implementação estão definidos nove níveis de adequação, que podem ser utilizados total ou parcialmente. É ainda definido um nível suplementar, designado por "Inadequado", que, quando seleccionado, impede o agendamento de determinada tarefa em determinada máquina.

A caixa de diálogo *Setups...* não foi activada, uma vez que nenhum dos algoritmos desenvolvidos faz uso dos tempos de preparação.

O menu *Objectivo* dá acesso à caixa de diálogo *Penalizações...* (figura ao lado), podem ser definidas as diversas penalidades a utilizar na matriz de custos do algoritmo de transportes. Assim, podem ser definidas as várias penalidades associadas aos vários níveis de adequação tarefa/processador, e podem ser definidas as penalidades associadas à violação de datas conclusão e, eventualmente, de datas de disponibilidade. Há ainda a possibilidade de

estabelecer uma *deadline* como um valor relacionado com a data de conclusão das tarefas, não permitindo uma violação desta data para lá de um determinado número de períodos de agendamento.
Para calcular a solução de uma instância devem accionar-se os comandos *Relax*, *Heurística 1* e *Heurística 2* do menu *Instância*. É possível observar a evolução da solução à medida que se executam os diferentes algoritmos.
A nível de solução, são apresentados ao utilizador três mapas diferentes. Um deles já foi utilizado para apresentar soluções nos capítulos anteriores. Trata-se do diagrama do tipo Gantt (figura ao lado). Um dos outros dois mapas apresenta alguns resultados calculados tarefa a tarefa e o outro apresenta resultados calculados máquina a máquina.

Dos resultados por tarefa calculados (figura seguinte), destacam-se a comparação entre as datas de disponibilidade e de início efectivo do processamento e a comparação entre as datas de conclusão e o final efectivo do processamento, permitindo calcular os atrasos. É ainda fornecida uma relação com o número de máquinas utilizadas para cada nível de adequação.

Quanto aos resultados por máquina, é calculado o número de preparações que cada máquina teve, o último período de agendamento e a percentagem de tempo utilizado da máquina.

7.4 Conclusão

Embora esta implementação tenha sido de grande utilidade no
dos algoritmos, é totalmente inadequada para utilização ao nível fabril, uma
que não está ligada ao sistema de informação da empresa e não tem um
linguagem adequado (por exemplo, um operador não trabalha com períodos
agendamento, mas com semanas, dias e horas). Uma interface dessa
que ser desenvolvida por profissionais da área da programação de sistemas
apoio à decisão. Por outro lado, a parte relativa aos algoritmos poderá,

eventualmente, necessitar de alguma afinação em termos de eficiência computacional.

Apesar de tudo, o próprio código dos algoritmos desenvolvidos poderia ser utilizado como ponto de partida para o desenvolvimento de uma aplicação profissional, uma vez que os algoritmos cumprem a função pretendida num período de tempo aceitável.

No próximo capítulo serão apresentados os resultados de testes realizados com a presente implementação.

Capítulo 8
TESTES COMPUTACIONAIS

Este capítulo é dedicado à análise dos resultados fornecidos pela implementação do modelo descrito nos capítulos anteriores. Esta análise tem como objectivo validar os algoritmos anteriormente desenvolvidos.

Numa primeira fase, será apresentada e discutida a metodologia seguida na realização dos testes computacionais. Numa fase posterior serão discutidos os resultados obtidos pela implementação efectuada. Entre outros pontos, será avaliado se os diversos algoritmos desenvolvidos se comportam da forma esperada, quanto a soluções obtidas.

8.1 Descrição da metodologia utilizada nos testes

Nos testes realizados foram utilizadas instâncias geradas aleatoriamente. A pela informação disponibilizada pela empresa onde o trabalho foi baseado, feita uma tentativa de aproximar as instâncias utilizadas nos testes às

reais do problema. No entanto, a geração de instâncias aleatórias de um problema com esta complexidade coloca algumas dificuldades.
Desde logo, o problema gerado não pode ser demasiado restritivo, pois a solução seria impossível. No extremo oposto, não se pode gerar instâncias com poucas restrições, pois não corresponderiam à realidade.

Alguns tipos de restrições sobre os quais não havia informação foram
Por exemplo, na matriz de adequações do problema, apenas foram
dois níveis, adequado e inadequado, devido à falta de informação existente
a quantidade de máquinas e tarefas onde os vários níveis de adequação

As instâncias geradas têm em comum as seguintes características:
– Número de processadores (100 processadores) e horizonte temporal (aproximadamente 90 períodos de agendamento);

– Cardápio de produtos: as ordens de encomenda utilizadas referiam-se a um cardápio de 10 referências possíveis. A referência de cada encomenda foi determinada aleatoriamente, sendo as probabilidades de cada referência iguais;

– Matriz de adequações: todas as instâncias geradas tiveram a mesma matriz de adequações. Nessa matriz está definido em que processadores pode ser agendada cada uma das 10 referências. Cada elemento da matriz corresponde a um de dois estados: adequado e inadequado. A matriz utilizada foi a seguinte:

Referência	Processador																			
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
R1											▨			▨						
R2												▨			▨					
R3													▨			▨				
R4												▨					▨			
R5											▨							▨		
R6												▨							▨	
R7	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨							▨
R8	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨						▨	
R9	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨							▨
R10	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		

Cada coluna da matriz corresponde a um conjunto de 5 processadores, representando cada linha da matriz uma das 10 referências utilizadas para gerar as instâncias. Os elementos da matriz a sombreado indicam as incompatibilidades. Por exemplo, a referência R1 não pode ser agendada no no conjunto de processadores C11.

A geração das instâncias foi feita recorrendo a números pseudo-aleatórios através de uma rotina própria da linguagem de programação utilizada. linguagem de programação permite a definição de uma "semente" aleatória, partir da qual os números serão gerados, é sempre possível voltar a gerar qualquer sequência de números aleatórios.

Foram utilizados dois algoritmos para a geração das instâncias. Cada algoritmo gerou 10 instâncias.

8.1.1 Algoritmo 1

As instâncias foram geradas tarefa a tarefa. Cada tarefa necessitou de 3 números aleatórios para ficar completamente definida.

O primeiro número aleatório definia a referência para a tarefa. Esta informação serve para gerar a matriz de adequações de acordo com o esquema já apresentado.

O segundo número aleatório definia o tamanho da tarefa. O tamanho da tarefa obedeceu a uma distribuição uniforme cujo mínimo era igual a 100 períodos de agendamento e o máximo igual a 600 períodos de agendamento. A partir deste valor definia-se a data de conclusão para obter uma taxa de ocupação geral aproximadamente igual 80% da capacidade máxima dos processadores.

Finalmente o terceiro número aleatório definia a data de disponibilidade. valor era igual a 1 (disponível no primeiro período de agendamento) primeira tarefa gerada e, para as restantes tarefas, um valor aleatório

uma distribuição uniforme de mínimo igual a 1 e máximo igual à data de conclusão da última tarefa gerada.

O tamanho da última tarefa era truncado, de forma a que a data de fosse aproximadamente igual ao número de períodos de agendamento para o tamanho da instância (90). Se o tamanho da tarefa, depois de fosse inferior ao tamanho mínimo, a tarefa não era considerada e o algoritmo terminava.

Nas instâncias geradas desta forma, existiam alguns desequilíbrios entre as cargas dos diversos processadores, originando soluções com atrasos significativos, não reflectindo deste modo a realidade.

8.1.2 Algoritmo 2

O segundo algoritmo é semelhante ao primeiro, excepto na definição de datas de conclusão e de disponibilidade e no tamanho máximo das tarefas.

As datas de conclusão passaram a ser determinadas, não só pela taxa de ocupação geral do sistema, mas também pela taxa de ocupação de conjuntos específicos de processadores. Assim, assegurou-se que os processadores dos conjuntos C19 e C20 não teriam uma taxa de ocupação superior a 80%. Fundamentalmente, apenas a referência R10 contribui para esta carga. Assegurou-se também que os processadores dos conjuntos C14 a C20 não teriam uma taxa de ocupação superior a 80%. Para esta carga contribuem as tarefas com as referências R7 a R10. A taxa de ocupação global do sistema também não deveria passar os 80%.

As datas de disponibilidade foram eliminadas, ficando todas as tarefas disponíveis no primeiro período de agendamento.

O tamanho máximo das tarefas foi reduzido para 300 períodos de agendamento.

Com estas regras pretendeu-se obter instâncias mais equilibradas, onde não houvesse tantos atrasos como nas instâncias geradas pelo algoritmo 1.

8.2 Apresentação e análise dos resultados

Nesta secção serão apresentados os resultados obtidos para variáveis que, de alguma forma indicam a qualidade dos algoritmos implementados. Como já foi dito, os resultados dizem respeito a 20 instâncias geradas aleatoriamente de acordo com os pressupostos explicitados nas secções anteriores.

8.2.1 Variáveis analisadas

As variáveis que foram objecto de análise foram o somatório dos valores de *lateness* ($\sum L_j = \sum \left(C_j - d_j\right)$), o somatório dos atrasos (*tardiness*) da solução ($\sum D_j = \sum max\left\{L_j, 0\right\}$), o número médio de máquinas utilizado em cada tarefa e o número médio de preparações por máquina.

As duas primeiras variáveis, $\sum L_j$ e $\sum D_j$, pretendem dar uma ideia do desempenho dos algoritmos em relação aos objectivos do problema. Uma vez que estamos perante uma versão simplificada do problema, em que o objectivo principal é minimizar os atrasos das tarefas, estas duas variáveis fornecem indicações sobre a qualidade da solução obtida.

O número médio de máquinas utilizadas por tarefa e o número médio de preparações por máquina fornecem indicações sobre a eficiência das heurísticas de redução de preparações.

Em todas as instâncias foram recolhidos os valores das diversas variáveis após a aplicação do algoritmo dos transportes (AT), após a aplicação da heurística 1 (H1) e após a aplicação da heurística 2 (H2). Desta forma pretendeu-se analisar a evolução da solução obtida à medida que se aplicavam os vários algoritmos.

Outra variável de interesse seria, eventualmente, o tempo de ocupação computacional dos diversos algoritmos. Devido a tratarem-se de tempos relativamente reduzidos (as instâncias mais demoradas correram durante cerca de 1 minuto num PC normal com um processador *Pentium II* a 350 MHz), estes não serão analisados. Por outro lado, não foram feitos grandes esforços no sentido de optimizar as implementações computacionais dos algoritmos. Assim, ao analisar esses tempos, estar-se-ia a analisar também a qualidade da implementação computacional.

8.2.2 Apresentação e análise dos resultados

Nesta secção apresentam-se os resultados obtidos nas 20 instâncias do problema geradas aleatoriamente. Esses resultados são apresentados separadamente para a primeira série de 10 instâncias (geradas com o algoritmo 1) e para a segunda série de 10 instâncias (geradas com o algoritmo 2). O motivo prende-se com a grande diferença que existe nos resultados obtidos para algumas variáveis, que não permite uma comparação.

A tabela seguinte mostra os resultados obtidos relativamente às duas primeiras variáveis em análise, o somatório dos valores de *lateness* ($\sum L_j$)e o somatório dos atrasos ($\sum D_j$).

Instância	ΣL_j			ΣD_j		
	Após AT	Após H1	Após H2	Após AT	Após H1	Após H2
1	6	6	6	6	6	6
2	29	9	9	29	20	20
3	78	75	75	78	75	75
4	34	34	34	34	34	34
5	52	36	36	52	36	36
6	129	85	85	139	100	100
7	79	69	69	79	69	69
8	195	139	139	195	155	155
9	81	50	50	90	77	77
10	75	62	62	75	62	62
Médias	**75.8**	**56.5**	**56.5**	**77.7**	**63.4**	**63.4**
11	0	0	-1	0	0	0
12	0	-3	-3	0	0	0
13	0	0	0	0	0	0
14	2	-9	-8	2	2	2
15	0	-10	-10	0	0	0
16	0	0	-1	0	0	0
17	0	0	0	0	0	0
18	0	0	0	0	0	0
19	0	-5	-5	0	0	0
20	0	0	0	0	0	0
Médias	**0.2**	**-2.7**	**-2.8**	**0.2**	**0.2**	**0.2**

Repare-se desde já na diferença significativa entre os valores exibidos na primeira série de 10 instâncias e os valores exibidos na segunda série. Esta diferença considerável deve-se aos diferentes métodos utilizados na geração das duas séries de instâncias, diferenças estas já explicadas anteriormente.
Tanto o valor de $\sum L_j$ como o valor de $\sum D_j$ sofrem uma redução após a aplicação de H1. Enquanto que a redução de $\sum L_j$ se explica redução de $\sum D_j$ é algo surpreendente. A redução de $\sum L_j$ deve-se ao heurística 1 minimizar $L_{\max}$ para cada processador, valor que não é pelo algoritmo dos transportes. A redução de $\sum D_j$ é uma aparente se o algoritmo de transportes produz soluções óptimas ao nível dos atrasos, como pode uma solução óptima ser melhorada por uma heurística?

A explicação é simples: a heurística 1 funciona, máquina a máquina, com base em datas de conclusão modificadas $d'_j = max(d_j, C_j)$. Se uma tarefa globalmente atrasada o não estiver numa máquina específica pode, após aplicação da heurística 1, ficar atrasada também nessa máquina, mas permitir desta forma, que tarefas eventualmente atrasadas nessa máquina deixem de o estar.

Esta forma de redução de atrasos, não prevista quando foi pensada a modificação das datas de conclusão a fornecer à heurística 1, tem vantagens e inconvenientes. Devido à estrutura da matriz de custos fornecida ao algoritmo de transportes, este tem tendência a fornecer soluções em que os atrasos se distribuem mais ou menos uniformemente pelas tarefas e pelas máquinas. Como resultado prático, tendencialmente, haverá várias tarefas e várias máquinas com pequenos atrasos, eventualmente recuperáveis na execução do plano de produção. Após a aplicação da heurística 1 ficarão menos tarefas atrasadas, mas com atrasos maiores e mais difíceis de recuperar, uma vez que se verificarão em várias máquinas.

Em implementações futuras, este ponto poderá ser tido em consideração. Note-se que para evitar que a heurística 1 produza este efeito, bastaria fornecer-lhe datas de conclusão das tarefas tais que $d'_j = max(d_j, C_{ij})$, conforme já foi demonstrado. A ideia será dar ao utilizador controlo sobre este aspecto da formação de uma solução. Após a aplicação da heurística 2 não há qualquer alteração nos valores de $\sum D_j$, o que faz todo o sentido, uma vez que esta heurística foi para reduzir o número de preparações a efectuar sem aumentar os $(\sum D_j)$ da solução.

Em termos de valores absolutos não pode ser feito nenhum comentário. Para isso seria necessário dispor de instâncias mais aproximadas da realidade, além da necessidade de possuir um termo de comparação com as actuais medidas de performance da empresa.
Na tabela seguinte são apresentados os resultados relativos ao número médio de máquinas utilizadas por tarefa e relativamente ao número médio de preparações por máquina, medidas que, em princípio, estarão correlacionadas.

Instância	N.º médio de máq. utilizadas por tarefa			N.º médio de preparações por máquina		
	Após AT	Após H1	Após H2	Após AT	Após H1	Após H2
1	54.71	54.71	54.48	11.03	10.50	10.45
2	43.05	43.05	41.24	10.33	8.82	8.46
3	36.39	36.39	35.94	6.78	5.84	5.75
4	45.10	45.10	42.90	8.63	8.23	7.79
5	40.85	40.85	39.80	7.99	6.98	6.73
6	34.26	34.26	33.22	8.97	8.14	7.97
7	33.00	33.00	32.33	7.56	6.79	6.58
8	38.11	38.11	36.79	8.34	6.80	6.60
9	39.17	39.17	36.52	9.23	8.41	7.92
10	42.95	42.95	41.65	8.22	7.92	7.66
Médias	**40.76**	**40.76**	**39.49**	**8.71**	**7.84**	**7.59**
11	44.72	44.72	44.42	15.83	15.10	14.99
12	43.82	43.82	42.82	14.87	14.00	13.66
13	27.04	27.04	27.00	6.69	6.55	6.54
14	31.66	31.66	29.60	11.75	10.08	9.36
15	30.41	30.41	30.13	9.47	8.73	8.64
16	32.43	32.43	29.57	12.18	11.01	9.95
17	21.56	21.56	21.50	2.74	2.69	2.68
18	24.21	24.21	23.93	6.50	6.29	6.21
19	34.32	34.32	33.58	14.25	12.04	11.76
20	32.87	32.87	31.39	12.10	11.49	10.93
Médias	**32.30**	**32.30**	**31.39**	**10.64**	**9.80**	**9.47**

Uma vez que as duas séries de 10 instâncias foram geradas de formas distintas, a análise será feita para cada série separadamente.
Conforme esperado, a heurística 1 faz reduzir o número de preparações efectuar. De facto, o número médio de preparações por máquina reduz-aplicação da heurística. Para a primeira série de instâncias verifica-se uma média de 0.87 preparações por máquina (se todos os tempos de forem iguais, houve uma redução de cerca de 9.93%). Para a segunda verifica-se uma redução média de 0.84 preparações por máquina (uma redução de cerca de 7.90%). Estes valores algo significativos indicam que a qualidade da solução melhorou após a aplicação da heurística 1, sem aumento dos atrasos.

A aplicação da heurística 2 teve efeitos na redução do número de máquinas utilizadas em cada tarefa e, por consequência, reduziu o número de preparações a efectuar. Assim, após a aplicação da heurística 2 cada tarefa utilizou, em média, menos 1.27 máquinas, para a primeira série de instâncias, e menos 0.91 máquinas para a segunda série de instâncias. Estes valores correspondem a uma redução dos tempos de preparação, no pressuposto que todas as preparações são iguais, de 3.21% para a primeira série de instâncias, e de 3.33% para a segunda série de instâncias.

Globalmente, a aplicação das duas heurísticas resultou na diminuição dos tempos de preparação em 12.83% para a primeira série de instâncias, e em 10.96% para a segunda série de instâncias. Estes valores, independentemente do tipo de instâncias utilizado, parecem significativos.

Saliente-se, como já foi referido, que a heurística 2 apenas tentou eliminar fragmentos com tamanho igual a 1 período de agendamento. Se este valor fosse aumentado, seria possível, em princípio, obter resultados melhores.

8.3 Conclusão

Devido a dificuldades diversas, não foi possível alargar e aprofundar os testes descritos neste capítulo. Assim, não foi possível testar o modelo em toda a sua extensão, tendo os testes ficado limitados a instâncias simplificadas. No entanto, os resultados obtidos apontam no sentido da validade do modelo apresentado.
Globalmente, não pode ser afirmado que as soluções finais obtidas são de boa ou má qualidade, pois falta um termo de comparação. Já quanto às duas heurísticas apresentadas, os resultados obtidos confirmam a sua qualidade, pois há uma materialização dos objectivos a que se propunham.
Na prática, para a implementação de um modelo deste tipo é necessária uma cuidadosa afinação dos parâmetros desse mesmo modelo, para que este bem adaptado à realidade do meio em que se vai inserir e assim se possa uma maior eficiência.

Capítulo 9
CONCLUSÕES E TRABALHO FUTURO

O presente capítulo divide-se em duas secções: uma secção dedicada às conclusões e uma secção dedicada à análise do trabalho futuro que poderá ser desenvolvido com base no presente trabalho.

9.1 Conclusões

O objectivo deste trabalho, em linhas gerais, era o desenvolvimento de um algoritmo que fornecesse soluções de boa qualidade num tempo aceitável. Embora as expressões "boa qualidade" e "tempo aceitável" tenham um significado subjectivo, uma vez que não foram quantificadas, os resultados apreciados na secção anterior permitem que possam aplicar ao trabalho desenvolvido.

A boa qualidade de uma solução define-se pelas características que aquela apresentar. No caso concreto, a principal especificação era o cumprimento datas de conclusão e o respeito pelas incompatibilidades produto/máquina.

demonstrado que o algoritmo utilizado, sempre que possível agenda de forma a obedecer às datas de conclusão e a respeitar as incompatibilidades estabelecidas.
Outro factor importante na qualidade de uma solução são os tempos de preparação. O algoritmo apresentado, depois de satisfazer as restrições mais importantes, faz um esforço no sentido de eliminar o maior número possível de preempções, com o objectivo de reduzir os tempos de preparação.

Ficou também demonstrado que outras características desejáveis das soluções de agendamento poderiam ser introduzidas através da correcta parametrização do modelo apresentado.
Por outro lado, os tempos de execução computacional dos algoritmos apresentados, mesmo com implementações computacionais não optimizadas, estão na ordem dos segundos, o que, tendo em conta o ambiente industrial em que irão ser implementados, parece perfeitamente aceitável.

Em face do exposto, pode concluir-se que as metodologias escolhidas para atacar o problema se revelaram adequadas e produziram bons resultados.

Aproveitando este trabalho de base, há um conjunto de trabalhos complementares com interesse para uma implementação fabril das metodologias. Este conjunto de trabalhos a executar será descrito na próxima secção.

9.2 Trabalho futuro

Tal como já foi dito, este trabalho pode servir de base a trabalhos complementares que concorram para aumentar a qualidade e utilidade do modelo apresentado.

Em primeiro lugar, alguns aspectos do problema que não foram modelo, devido ao seu grau de dificuldade e à sua complexidade deveriam repensados à luz da relação valor/custo. Ou seja, se se considerar que, por

exemplo, a proximidade entre teares utilizados numa determinada tarefa é muito importante, poder-se-ia investir mais algum esforço na procura de uma solução para o problema e na adaptação do modelo desenvolvido. Ainda a nível do modelo desenvolvido há aspectos em que poderá ser compensador o investimento de mais algum esforço. Um exemplo disso é a melhoria do desempenho computacional da heurística H2 que, para instâncias um pouco maiores ou para uma maior profundidade, poderá vir a ter tempos de execução incómodos ou inaceitáveis.

Os algoritmos apresentados poderiam ser utilizados para estabelecer datas de conclusão para as tarefas mediante uma adaptação simples. Quando as encomendas são combinadas é possível negociar os prazos de entrega. Conhecendo de antemão e de forma rigorosa a capacidade do sistema, possivelmente, poder-se-iam oferecer prazos de entrega mais curtos e haveria um maior equilíbrio entre as cargas dos diversos processadores. A adaptação a fazer consiste em experimentar o agendamento com uma determinada data e verificar se este é possível e em que condições.

O subsistema de agendamento poderia também ser integrado num sistema mais vasto do tipo MRP. Nesta caso a informação constante do plano de agendamento poderia ser utilizada no planeamento das secções a montante, nomeadamente na compra de matérias-primas, e a jusante, no planeamento das operações nas secções subsequentes.

Outro trabalho a desenvolver é a implementação a nível operacional do modelo apresentado. Este trabalho é bastante complexo porque, embora os algoritmos fundamentais estejam desenvolvidos, é necessário integrar o modelo no sistema de informação e no sistema de apoio à decisão da empresa.

Depois da implementação terá de ser feita, necessariamente, a modelo. Esta parametrização consiste, tal como foi discutido na

modelo, em atribuir pesos a diferentes factores do modelo, de modo a conseguir soluções mais de acordo com as necessidades da empresa.

9.3 Considerações finais

Atendendo à especificidade do problema em análise, a abordagem que foi utilizada permitiu obter um conjunto de metodologias suficientemente genéricas e flexíveis, de modo a poderem ser utilizadas fora da empresa que serviu de base ao seu desenvolvimento.

De facto, podem imaginar-se variadas situações produtivas onde as principais condições que dão forma ao problema analisado se repetem, permitindo a aplicação das técnicas aqui descritas.

Por outro lado, na literatura sobre planeamento operacional os problemas existe preempção e possibilidade de processamento paralelo são pouco em relação a outros tipos de problemas. Igualmente, existe pouca literatura problemas com tempos de preparação. Este trabalho, embora inserido muito estudada, incide sobre aspectos menos estudados. Este facto, faz com o presente trabalho seja bastante inovador quanto ao assunto objecto de

BIBLIOGRAFIA

[1] AHUJA, RAVINDRA K.; MAGNANTI, THOMAS L.; ORLIN, JAMES B. – *Network flows: theory, algorithms, and applications*, Englewood Cliffs, Prentice Hall, 1993.
[2] BERTSEKAS, DIMITRI P.; TSENG, PAUL – *The relax codes for linear minimum cost network flow problems*, Annals of Operations Research 13, p. 125-190, 1988;
[3] BLAZEWICZ, JACEK; ECKER, KLAUS H.; SCHMIDT, GÜNTER; WEGLARZ, JAN – *Scheduling in computer and manufacturing systems*, Berlim, Springer-Verlag, 1994.
[4] BRUCKER, PETER – *Scheduling algorithms*, Berlim, Springer-Verlag, 1995.
[5] BRUCKER, PETER – *Complexity Results for Scheduling Problems*, Universidade de Osnabrueck,
http://www.mathematik.uni-osnabrueck.de/research/OR/class/;
[6] CHENG, T. C. E.; SIN, C. C. S. – *A state-of-the-art review of parallel-machine scheduling research*, European Journal of Operational Research 47, p. 271-292, North-Holland, 1990;
[7] GAREY, MICHAEL R.; JOHNSON, DAVID S. – *Computers and Intractability*, São Francisco, W. H. Freeman and Company, 1979;
[8] GRAHAM, R.; LAWLER, E.; LENSTRA, J. K.; RINNOOY KAN, A. – *Optimization and Approximation in Deterministic Sequencing and Scheduling: a survey*, Annals of Discrete Mathematics, Vol. 5, North-Holland, 1979;
[9] HOOGEVEEN, J. A., LENSTRA, J. K.; VELDE, S. L. – *Sequencing and Scheduling: an annotated bibliography* in *Annotated bibliographies in combinatorial optimization*, eds. Mauro Dell' Ammico, Francesco Maffioli, Silvano Martello, Nova Iorque, John Wiley & Sons, 1997;
[10] HORN, W. A. – *Some simple scheduling algorithms*, Naval Res. Logist. Quart., 21, 1974;
[11] LENSTRA, J. K.; RINNOOY KAN, A. – *Sequencing and Scheduling: an annotated bibliography* in *Annotated bibliographies in combinatorial optimization*, eds. M. O'hEigeartaigh, J. K. Lenstra, A. Rinnooy Kan, Amsterdão, John Wiley & Sons, 1985;
[12] LENSTRA, J. K.; RINNOOY KAN, A.; BRUCKER, PETER – *Complexity of machine scheduling problems*, Annals of Discrete Mathematics, Vol. 1, North-Holland, 1977;

[13] MCNAUGHTON, R. – *Scheduling with deadlines and loss functions*, Management Science, Vol. 6, p. 1-12, 1959;
[14] MORTON, THOMAS E.; PENTICO, DAVID W. – *Heuristic scheduling systems: with applications to production systems and project management*, Nova Iorque, John Wiley & Sons, 1993;
[15] NEMHAUSER, G.; WOLSEY, L. – *Integer and combinatorial optimization*, John Wiley & Sons, 1988;
[16] PINEDO, MICHAEL– *Scheduling: theory, algorithms, and systems*, Englewood Cliffs, Prentice Hall, 1995.
[17] *Journal of Scheduling*, John Wiley & Sons;