AFETAÇÃO DE RECURSOS EM SISTEMAS DE PRODUÇÃO

André Borges Guimarães Serra e Santos

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

2013

AFETAÇÃO DE RECURSOS EM SISTEMAS DE PRODUÇÃO

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores – Sistemas e Planeamento Industrial

Candidato: André Borges Guimarães Serra e Santos, Nº 1070953, 1070953@isep.ipp.pt
Orientação científica: Doutora Ana Maria Dias Madureira Pereira, amd@isep.ipp.pt

Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
17 de novembro de 2013

Agradecimentos

Aproveito a oportunidade para deixar os meus sinceros agradecimentos à Prof. Ana Madureira Pereira, pela oportunidade proporcionada para desenvolver este trabalho, numa área em que tenho tanto interesse, bem como por toda a disponibilidade, motivação e apoio que me foram essenciais para a elaboração desta tese.

Deixo também, um especial agradecimento a Prof. Ana Viana e a Prof. Luísa Hoffbauer pelo auxilio e pelos apontamentos que me facultaram.

Uma palavra para os meus colegas e amigos.

Para toda a minha família.

Resumo

O problema do escalonamento, por ser um dos factores fundamentais na tomada de decisão para uma boa gestão das operações, tem sido alvo de um amplo estudo, tanto na sua componente teórica como na sua componente prática. A importância de um escalonamento correto das operações é preponderante, quando as pequenas diferenças, em termos de tempos de produção, podem ter um grande impacto na competitividade da organização.

Em muitas unidades produtivas, existem máquinas capazes de realizar as mesmas operações com diferentes desempenhos. Isto pode dever-se à necessidade de flexibilizar os recursos ou mesmo a uma atualização da capacidade produtiva. Embora os problemas de máquinas diferentes em paralelo tenham sido alvo de um vasto estudo, muitos deles não são passíveis de ser resolvidos através de métodos exatos. O problema de minimização do *makespan* ($R_m||C_{max}$), é *NP-hard*, sendo habitualmente abordado através de heurísticas.

Entre as heurísticas utilizadas em problemas de minimização do *makespan* em máquinas diferentes em paralelo, é possível identificar duas filosofias de afectação: a que utiliza os tempos de processamento para alocar as tarefas e a que utiliza as datas de conclusão.

Nesta dissertação, pretende-se dar uma contribuição para a resolução do problema de afectação de recursos em sistemas de produção. Para tal, foram propostas as heurísticas OMTC 3 e Suffrage One. A contribuição consiste na proposta de versões híbridas e modificadas das heurística MCT e Suffrage, uma vez identificadas várias características que podem limitar o seu desempenho, como o facto da heurística MCT alocar as tarefas numa ordem aleatória ou a heurística Suffrage alocar mais que uma tarefa por iteração.

Finalmente, procedeu-se à realização de testes computacionais, para avaliar o desempenho das heurísticas propostas. Os testes realizados permitiram concluir que a heurística OMTC 3 apresentou um melhor desempenho que a heurística MCT.

Palavras-Chave

Escalonamento, *Makespan*, Máquinas Diferentes em Paralelo, OMCT, Suffrage One.

Abstract

Schedule problems, both theoretical or real, have been researched extensively . The correct schedule of operations is fundamental when a small difference in processing times can have a major impact on the competiveness of an organization.

In several production units there are similar machines with different performance capabilities. Scheduling problems in parallel machines have been deeply studied and many are too complex to be solved exactly. The unrelated parallel machines makespan minimization problem ($R_m||C_{max}$) is NP-hard and is usually solved using heuristics.

Between the heuristics used in unrelated parallel machines makespan minimization problems, there are two different approaches. Those that use the processing time to allocate tasks and those that use the completion time.

This dissertation objective is to contribute to the resolution of resources allocation problems in production systems. Two new heuristics are proposed, OMCT 3 e Suffrage One, based on the performance limitation of the MCT and Suffrage heuristics. The first allocates tasks in random order while the second allocates several tasks in one iterations.

The two proposed heuristics were evaluated in the computational study. The OMCT 3 heuristic had a better performance than the MCT heuristic in the computational study.

Keywords

Scheduling, Makespan, Unrelated Parallel Machines, OMCT, Suffrage One.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

ATC	-	Apparent Tardiness Cost
CP	-	Critical Path
EDD	-	Earliest Due Date
FIFO	-	First In First Out
JIT	-	Just In Time
LFJ	-	Least Flexible Job
LNS	-	Largest Number of Successors
LPS	-	Longest Processing Time
LRPT	-	Longest Remaining Processing Time
MCT	-	Minimum Completion Time
MET	-	Minimum Execution Time
MRP	-	Material Requirements Planning
QFD	-	Quality Function Deployment
OMCT	-	Ordered Minimum Completion Time
OTED	-	One-Touch Exchange of Die
PDP	-	Plano Diretor de Produção
PE	-	Plano Estratégico
PEST	-	Political, Economic, Social, and Technological

PIC	-	Plano Industrial e Comercial
SMED	-	Single-Minute Exchange of Die
SPT	-	Shortest Processing Time
SRPT	-	Shortest Remaining Processing Time
SRPT-FM	-	Shortest Remaining Processing Time on Fastest Machine
SWA	-	Switching Algorithm
SWOT	-	Strengths, Weaknesses, Opportunities and Threats
WDSPT	-	Weighted Discounted Shortest Processing Time
WSPT	-	Weighted Shortest Processing Time

1. INTRODUÇÃO

Com a revolução industrial e a operacionalização das primeiras fábricas, surgiu a necessidade do escalonamento das operações. O escalonamento é uma função de decisão, que acrescenta o detalhe final às ordens de produção. Se inicialmente era apenas necessário decidir quando é que uma ordem devia começar a ser executada, com o aumento de complexidade das fábricas passou a ser necessário decidir com que recursos e por que ordem as operações seriam executas, isto é, como é que as tarefas vão ser afectadas aos recursos disponíveis à produção e qual a sequência em que vão ser processadas.

A complexidade dos sistemas produtivos e as restrições a si associadas, bem como o elevado número de tarefas a serem executadas, podem tornar estes problemas extremamente complexos. Essa complexidade levou a uma divisão nos métodos de resolução utilizados num problema de escalonamento, entre: Métodos Enumerativos e Métodos Heurísticos. Os primeiros, embora exatos, são por vezes impossíveis de calcular em tempo útil, enquanto os métodos heurísticos apresentam um compromisso entre a solução e o tempo disponível para obter essa solução.

Com o estudo do escalonamento foram criados modelos para os problemas que apresentam características semelhantes. A divisão mais habitual é feita pela forma como os recursos estão dispostos, pois é fácil encontrar semelhanças nas abordagens ao problema de escalonamento em dois sistemas produtivos a operar com as máquinas dispostos na mesma tipologia. Outras maneiras de detalhar os modelos são as restrições dos sistemas produtivos e os objectivos do escalonamento.

Um caso particular dos problemas em máquinas paralelas, são aqueles onde as máquinas apresentam características diferentes. Este é um problema interessante, tanto do ponto de vista industrial, pois pode caracterizar uma situação em que foi realizado um *upgrade* à capacidade produtiva com uma máquina diferente das existentes, como do ponto de vista informático, onde pode descrever um sistema de computação partilhada (*Grid Computing*), em que os computadores, habitualmente, não têm todos as mesmas características.

Um dos propósitos mais habituais num problema de escalonamento, é a diminuição do *makespan*, isto é, a redução do tempo desde do inicio de execução de uma ordem até à sua conclusão. Podem existir muitos outros propósitos para o escalonamento, dependendo dos objectivos tecnológicos e económicos da organização. Estes podem ir desde a redução do tempo médio de execução até à redução dos tempos de atraso.

1.1. OBJECTIVOS

Os problemas de escalonamento têm sido tradicionalmente abordados como problemas de optimização sujeitos a restrições, cujos elementos básicos são as máquinas e as tarefas. Pode considerar-se que o escalonamento é constituído por duas fases: a afectação ou atribuição das operações das tarefas aos recursos do sistema e o respectivo sequenciamento e calendarização. Estas fases podem ser tratadas separadamente ou duma forma integrada, dependendo do tipo e dimensão do sistema em questão e da estratégia de escalonamento.

Pretende-se o desenvolvimento de um módulo de afectação de operações a recursos ou máquinas, através de:

- Estudo das principais características e restrições dos problema do escalonamento;
- Análise dos problemas de minimização do *makepsan* em ambientes de máquinas paralelas, de máquinas paralelas com interrupções (*preemptions*) e em máquinas diferentes (*unrelated*) em paralelo;
- Estudo comparativo das heurísticas utilizadas, em problemas de minimização do *makepsan* em máquinas diferentes em paralelo;
- Desenvolvimento e implementação de heurísticas para a minimização do *makespan* em ambientes de máquinas diferentes em paralelo;
- Estudo computacional.

1.2. ORGANIZAÇÃO DO RELATÓRIO

O relatório encontra-se divido em sete capítulos:

No primeiro capítulo é realizada uma pequena introdução ao trabalho desenvolvido e são definidos os objetivos que se pretendem atingir com este trabalho de mestrado.

No segundo capítulo o problema do escalonamento é contextualizado como função de decisão da gestão das operações. É analisada a interação entre o problema de escalonamento e as outras fases do planeamento da produção.

No terceiro capítulo são apresentados os conceitos que sustentam o trabalho e é definido o problema do escalonamento, as suas características e restrições. É também, realizado um resumo dos problemas de escalonamento mais habituais.

No quarto capítulo é descrito o problema de minimização do *makespan* em máquinas paralelas e realizada a comparação das heurísticas mais utilizadas para minimização do *makespan* em problemas de máquinas diferentes em paralelo.

No quinto capítulo são analisadas as diferenças entre as heurísticas propostas. É explicado o método de implementação das heurísticas que vão ser analisadas no estudo computacional.

O sexto capítulo descreve o estudo computacional e apresenta a análise dos resultados.

Finalmente, no sétimo capítulo são apresentadas as conclusões do trabalho realizado e identificados possíveis trabalhos futuros.

2. GESTÃO DAS OPERAÇÕES

Neste capítulo, serão descritos os conceitos necessários para a compreensão e resolução de problemas de Escalonamento da Produção, dando especial atenção à fase da afectação. De maneira a contextualizar os problemas a abordar, serão também descritos os principais conceitos da Gestão das Operações.

Serão apresentados dois métodos para classificar sistemas produtivos.

2.1. Introdução

O departamento de Operações/Produção, é um dos departamentos que habitualmente desempenha funções primárias dentro de uma organização.

Entende-se como função primária, aquela que é responsável pelo cumprimento dos principais objectivos a que as organizações se propõem, isto é, é aquela que tem um impacto direto na sua capacidade competitiva. Tal como, o departamento Comercial (que encontra ou desenvolve mercados para os produtos ou serviços disponibilizados pela organização), o departamento Financeiro (que permite à organização financiar-se), o departamento Desenvolvimento de Produto (responsável pelo desenvolvimento de novos produtos ou serviços), também o departamento de Operações/Produção exerce uma função indispensável para atingir os objectivos da organização ao transformar os *inputs*, em *outputs* disponíveis ao público.

É impossível falar em produção, sem mencionar o conceito de valor acrescentado. Assim é possível definir produção como a função de transformação de recursos, como matérias-primas ou trabalho, em produtos ou serviços de valor superior aqueles que os geraram. O processo de transformação levado a cabo pelo departamento de Operações/Produção encontra-se representado abaixo, na figura 1.

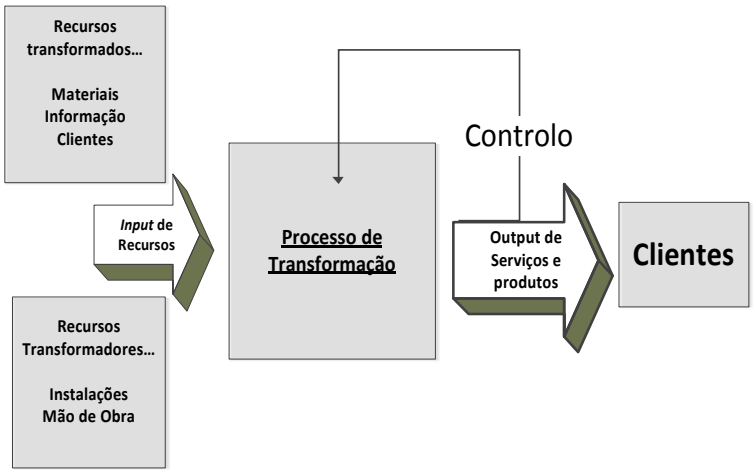

Figura 1 - Função das Operações / Produção [3]

Um sistema produtivo terá que obedecer a alguns princípios básicos e assim deverá ter um objectivo bem definido; não deverá operar de forma isolada, mas inserido no contexto organizacional; deverá existir um fluxo de informação durante todo o processo de forma a garantir o controlo do sistema [1] [2] [3].

2.2. Conceitos de produtos e serviços

Durante os anos 70, devido ao crescimento exponencial da indústria de serviços, o termo Gestão da Produção foi alterado para Gestão das Operações, de forma a englobar o sector dos serviços em rápido crescimento. Atualmente, nos países desenvolvidos, o número de postos de trabalho gerados pela indústria dos serviços é muito superior aos gerados pela indústria produtiva.

A principal diferença entre um produto e um serviço é a tangibilidade do produto e a intangibilidade dos serviços. Assim, e devido à intangibilidade dos serviços, estes não podem ser armazenados para um consumo posterior; este facto é muito relevante pois irá impor limites à maneira como a organização vai operar e como deverá afectar os seus recursos produtivos.

Embora a distinção entre serviço e produto (figura 2) possa parecer clara, em termos práticos a fronteira é muito mais difusa. Primeiro porque a maioria das organizações produzem tanto produtos como disponibilizam serviços, mas também porque se pode argumentar que o cliente ao adquirir um produto, está a comprar o serviço de produção desse produto.

Figura 2 - Distinção entre Produto e Serviço [4]

O crescimento da sociedade de informação veio complicar ainda mais a distinção entre serviço e produto, particularmente, quando um produto adquirido em formato físico é classificado como produto, mas quando adquirido em formato digital já é considerado um serviço [1] [2] [4].

2.3. Planeamento das operações

O planeamento constitui uma das mais importantes atividades na gestão de operações. De facto, o bom planeamento, nas suas diferentes fases, permite assegurar a eficácia da organização em atingir os seus objetivos e a eficiência no uso dos seus recursos. A eficácia e eficiência de um sistema são valores que não podem ser desassociados para avaliação da organização, isto é, a existência de uma sem o outra não permite alcançar os objectivos a que a organização se propõem.

É impossível falar em planeamento sem o recurso a ferramentas de previsão. De nada serviria planear, se não houvesse a mínima expectativa de que as condições para as quais o plano foi elaborado viessem a acontecer. Previsões de procura, de oportunidades, de financiamento e tecnológicas, permitem elaborar e suportar com antecedência uma estratégia para a organização.

Assim, o plano visa detalhar os objectivos a que a organização se propõe, definir os métodos de controlo e as ações a tomar para atingir esses objectivos. O planeamento deverá seguir técnicas de progressivo detalhe (figura 3) e ser modificado sempre que as condições para o qual foi preparado forem alteradas. As decisões a longo prazo, que influenciam a direção de toda a organização, deverão ser gerais e englobam as decisões de posicionamento de mercado, especificações de produtos, *layouts* e capacidades. O planeamento a curto prazo deverá ser o mais detalhado possível, definindo todas as ações a tomar para atingir as metas de produção propostas [1] [5].

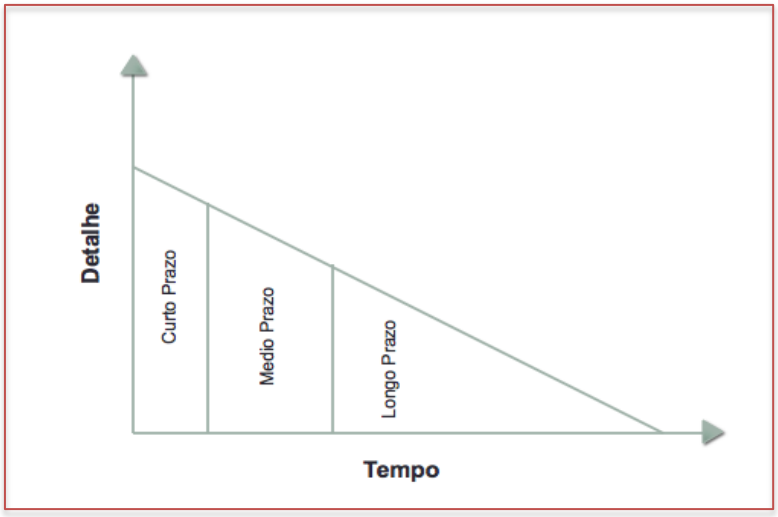

Figura 3 - Detalhe no Planeamento [1]

2.3.1. Plano Estratégico (PE)

É o planeamento de longo prazo, que irá guiar a organização e determinar os seus objectivos. Irá também definir os produtos que irão ser disponibilizados, caracterizar o mercado onde a organização vai operar, estabelecer o seu posicionamento e definir as ações competitivas [5].

Para preparação do plano estratégico é necessário considerar tanto os factores internos da organização, como os factores do meio onde está inserida.

Identificam-se como factores internos que deverão ser utilizados como *inputs* [5]:

- **Ativos Existentes** - Fábricas e equipamentos;
- **Clientes** - Relações com os clientes;
- **Fornecedores** - Relações com os fornecedores;
- **Propriedade Intelectual** - Direitos e patentes da organização;
- **Recursos Humanos** - Número de colaboradores e as suas capacidades;
- **Situação Financeira** - Capital existente e possibilidades de financiamento;
- **Tecnologia** - Capacidade tecnológica da organização.

Identificam-se como factores externos que serão utilizados como *inputs* [5]:

- Condições Económicas;
- Condições do Mercado;
- Condições Politicas;
- Condições Sociais;
- Condições Tecnológicas.

Para preparar o plano estratégico poderão ser aplicadas varias técnicas, como: Análise SWOT (*Strengths, Weaknesses, Opportunities and Threats*); Análise PEST (*Political, Economic, Social, and Technological*); Modelo de Kano ou o QFD (*Quality Function Deployment*).

Este plano vai servir como linha orientadora da toda a política interna, e o planeamento de mais curta duração vai utilizar o plano estratégico como o seu principal *input*. A importância da avaliação dos factores externos e das características internas da organização não pode ser menosprezada, pois os objectivos a longo prazo devem ser adequados à realidade, sob pena de se criarem expectativas não alcançáveis, que podem colocar em causa toda a capacidade operacional da organização [1] [5].

2.3.2. Plano Industrial e Comercial (PIC)

O Plano Industrial e Comercial situa-se imediatamente abaixo do Planeamento Estratégico e tem como principais intervenientes os responsáveis pelos departamentos de produção, de logística e comercial. Como se trata de um planeamento essencial para enquadrar as atividades, é fundamental a participação dos três departamentos mencionados.

A elaboração deste tipo de planeamento permite analisar toda a atividade da organização. Para fazer essa análise é necessário definir famílias de produtos que possam ser estudados mais facilmente. É normal impor limites ao número de famílias de produtos que serão analisadas. Estas não deverão ser muito numerosas pois pode perder-se a perspetiva global que este tipo de planeamento oferece e iria aumentar de uma maneira significativa o esforço necessário para a sua preparação. Normalmente são consideradas, no máximo, 20 famílias de produto. O seu horizonte deverá depender não só do prazo global de disponibilização do produto mas também dos prazos esperados para que medidas corretivas possam ser implementadas.

O Plano Industrial e Comercial vai definir as encomendas que serão feitas ao departamento de produção por cada família de produtos. Os resultados deverão ser apresentados em toneladas, euros ou qualquer outra escala que permita englobar todos os artigos pertencentes a essa família de produtos.

Este tipo de planeamento deverá permitir à organização decidir a alocação de recursos e antecipar problemas de inadequação da carga à capacidade da unidade de produção. Para a sua preparação são necessárias conhecer, as unidades vendidas, as unidades produzidas e os produtos em *stock*. É preciso ter em conta valores passados, quanto foi produzido no período passado ou qual é o nível de stock atual, bem como previsões para o futuro, quanto se espera produzir no próximo período ou quais são as previsões para as vendas.

É possível optar por adequar a produção à procura ou manter um nível de produção constante aceitando as variações no *stock*. É neste contexto que os *stocks* começam a apresentar-se como tema inevitável. O recurso aos *stocks* é um compromisso entre diferentes interesses, o do departamento comercial que deseja ter produtos em *stock* para satisfazer os clientes e do departamento financeiro que deseja diminuir os custo de armazenamento (Custo de Oportunidade, Custo de Manutenção, Custo de Armazenamento, etc.). As políticas de *stock* são um dos temas mais difíceis de tratar. É impossível afirmar que um política de *stocks* é melhor que outra, devido as especificidades de cada organização.

Depois de se ter planeado a carga para um determinado período de tempo, é necessário verificar se existe capacidade para a executar.

No caso da carga ultrapassar a capacidade, será necessário optar entre duas soluções: o aumento da capacidade ou diminuição da carga. Geralmente é preferível o aumento da capacidade, pois a carga representa clientes e o valor acrescentado a eles associado. O aumento da capacidade torna-se possível, contratando mais pessoal ou equipamento, recorrendo a horas extraordinárias ou à subcontratação. O horizonte alargado do plano industrial e comercial, deverá permitir implementar as medidas corretivas para a adequação da capacidade à carga esperada [1] [2].

2.3.3. Plano Diretor de Produção (PDP)

A natureza genérica do Plano Industrial e Comercial, onde os produtos são tratados em famílias, leva à necessidade de um planeamento mais fino, que permita quantificar o número de artigos a produzir. O Plano Diretor de Produção é responsável por transformar o Plano Industrial e Comercial em produtos e representa um dos principais canais de comunicação entre os departamentos funcionais da organização. Isto é, vai permitir calendarizar as quantidades a produzir de cada produto, em cada período.

O Plano Diretor de Produção tem um alcance temporal que nunca deve ser menor do que todos os prazos acumulados, respeitando as regras de precedência, do fabrico de todos os componentes que constituem o produto. A calendarização do PDP, deve contemplar intervalos menores que o Plano Industrial e Comercial e ser revisto mais frequentemente.

Para preparar o Plano Diretor da Produção são necessários obter informação do:

- Departamento de marketing, quanto se espera vender;
- Departamento comercial, quantos artigos foram encomendados;
- Departamento de logística, quantos artigos existem no armazém;
- Departamento de produção, quantos produtos se encontram em curso de fabrico.

Torna-se ainda necessário:

- Definir o tamanho de cada lote a produzir (L);
- Estabelecer o *Lead Time* (D) de cada produto.
- Dimensionar o stock de segurança (SS), destinado a assegurar um bom serviço perante um consumo incerto;

O resultado deste tipo de planeamento será um mapa detalhado dos produtos que deverão ser produzidos em cada período. Mostra-se um exemplo de um PDP simplificado, onde apenas são consideradas as necessidades brutas na tabela 1 [1] [2] [3].

Tabela 1 - Plano Diretor de Produção Simplificado [4]

Produto: Cabine de CDs				Lead Time: 1 semana								
Tamanho do Lote: 100				Stock Inicial: 80								
	1	2	3	4	5	6	7	8	9	10	11	12
Necessidades Brutas:	25	25	25	25	30	30	30	30	35	35	35	35
Disponível Previsional:	55	30	5	80	50	20	90	60	25	90	55	20
PDP:				100			100			100		

2.3.4. Cálculo das necessidades líquidas

Antes de descrever qual a utilidade e o procedimento, para o Cálculo das Necessidades Liquidas torna-se necessário explicar a sua origem. O conceito do MRP (*Material Requirements Planning),* apareceu quando Joseph Orlicky, notou as diferenças entre as Necessidades Independentes e as Necessidades Dependentes [5]. As Necessidades Independentes referem-se às necessidades exteriores à organização, ou seja, à procura por produtos finais. Esta necessidade, embora possa ser influenciada, pelo Departamento Comercial e de Marketing, é independente da vontade da organização. As Necessidades Dependentes referem-se às necessidades geradas no interior da própria organização, ou seja, correspondem a necessidades em componentes ou produtos em curso de fabrico, que devem ser satisfeitas para que as Necessidades Independentes possam ser cumpridas.

Compreendida essa diferença, Joseph Orlicky expressa-o naquilo que é hoje conhecido como o Principio de Orlicky:

> **Teorema 1** As Necessidades Independentes podem ser estimadas a partir das previsões. As Necessidades Dependentes, pelo contrário, devem ser calculadas [5].

Finalmente é possível compreender em que se baseia o Cálculo das Necessidades Liquidas, ou seja, no exemplo da tabela 1, as previsões de venda de Cabines de CDs são baseadas em previsões de procura mas o cálculo das ordens de componentes que constituem as Cabines de CDs devem ser calculadas.

Assim é possível afirmar que o Cálculo das Necessidades Liquidas é a especificação realizada a partir das Necessidades Independentes e das Necessidades Dependentes. Este cálculo vai permitir conhecer os prazos e os volumes de aprovisionamento, bem como as ordens a lançar de acordo com as datas de entrega de produtos acabados especificada pelo Plano Diretor de Produção.

Para realizar o Cálculo das Necessidades Líquidas é necessários definir para cada produto:

- O valor do stock de segurança;
- Os prazos de fabrico dos produtos;
- Os prazos de aprovisionamento de matérias primas ou produtos comprados;
- As nomenclaturas (os componentes discriminados em precedência e quantidade) que constituem o produto;
- O número de produtos ou matérias primas em armazém; O número de produtos em curso de fabrico; O tamanho dos lotes.

O resultado do Cálculo das Necessidades Líquidas será: As ordens de fabrico a lançar e os aprovisionamentos previstos. Um exemplo do Cálculo das Necessidades Liquidas poderá ser analisados na tabela 2.

Tabela 2 - Cálculo das Necessidades Liquidas [4]

Produto: Topo da Cabine	Lead Time: 3 Semanas			Nível Superior: Cabine de CDs							
Tamanho do Lote: 144	Inventário Inicial: 120			Nível Inferior: /							
	1	2	3	4	5	6	7	8	9	10	11	12
Necessidades Liquidas	0	0	100	0	0	100	0	0	100	0	0	0
Recepções Agendadas												
Disponível Previsional	120	120	20	20	20	64	64	64	108	108	108	108
Ordens Lançadas			144			144						

Como é possível constatar o Cálculo das Necessidades Líquidas vai, a partir da nomenclatura do produto, definir quais serão as ordens a lançar para cumprir as necessidades brutas do artigo. A linha das necessidades líquidas deve ser preenchida de acordo com as necessidades brutas, tendo em conta o número desse tipo de componentes que deverão estar presentes no produto, bem como o seu *Lead Time*. Sabendo quais são as necessidades liquidas para o componente, a tabela é simples, requerendo apenas que sejam respeitados os tempos de produção e os stocks existentes [1] [4].

2.4. *Just in Time* (JIT)

Nas secções anteriores o planeamento e o controlo da produção foram analisados no contexto do MRP. No entanto, durante os anos 80, uma nova filosofia baseada em técnicas usadas pela Toyota desde os anos 50, começou a ser considerada a nível mundial. A nova abordagem, designada por JIT (*Just In Time*), tinha como principio, produzir apenas aquilo que ia ser vendido, e apenas quando era necessário. De facto, a filosofia desenvolvida pela Toyota, contrariava os principais princípios usados pelas empresas ocidentais. Se antes se produzia para posteriormente vender, com o *Just In Time*, vendia-se primeiro para só depois produzir aquilo que se sabia que ia ser consumido [5].

O *Just In Time* pode ser considerado uma filosofia, pois para a sua correta implementação é necessário recorrer a várias técnicas que levam a mudanças estruturais na organização. O principal objectivo é a redução de custo, que deverá ser apenas o indispensável para acrescentar valor ao produto, ou seja, reduzir todo o desperdício.

De maneira a reduzir os custos há que resolver uma série de problemas organizacionais, tais como: organização deficiente; stocks excessivos; prazos de produção demasiado longos; falta de flexibilidade; falta de fiabilidade; pessoal com formação insuficiente.

A estes problemas estão normalmente associados a: desorganização das oficinas; implementações incorretas; elevados tempos de *setup*; problemas de qualidade; avarias sucessivas; falta de fiabilidade dos fornecedores; má formação do pessoal. Os problemas citados representam um entrave considerável à eficiência das organizações, pondo em causa a sua competitividade em mercados mais voláteis. As principais consequências dessa falta de competitividade podem refletir-se em: atrasos nas entregas; desperdício; falta de fiabilidade; falta de motivação; produtos defeituosos; stocks elevados.

Para a correta implementação do JIT foram desenvolvidos vários métodos. Em 1958, a Toyota Motor Company implementou nas suas linhas de produção um sistema de informação, por etiquetas, a que chamou “Kanban”. Esse sistema de produção, alterava a maneira como as ordens de fabrico eram lançadas, de maneira a assegurar que apenas era produzido o que era necessário. Os postos de trabalho a montante iriam apenas produzir aquilo que é solicitado pelos postos a jusante.

Esta é uma alteração muito significativa na maneira com a produção vai decorrer, pois se no contexto do MRP a produção funciona em *Push* (as ordens são empurradas através dos sistema), o Kanban ira colocar a produção a funcionar em *Pull* (as ordens são puxadas pelo do sistema), ou seja, as ordens de fabrico serão sempre lançadas com base nas necessidades do posto imediatamente a jusante [5].

Para a implementação do *Just In Time*, é necessário: reduzir os tempos de setup, através de métodos como o SMED e o OTED; introduzir politicas de controlo de qualidade, como o Six Sigma, desenvolvido pela Motorola; reorganizar o espaço de produção, seguindo os 5S (*Sort*, *Straighten*, *Sweep*, *Standardize* e *Sustain*). A relação com os fornecedores também deverá sofrer alterações de maneira a garantir a qualidade dos produtos recebidos e que os produtos são entregues nas condições/prazos certos [1] [5].

2.5. Funções e objectivos

O planeamento é uma atividade pré-produtiva que determina quais serão as necessidades da empresa. Como o planeamento é baseado em previsões de natureza incerta torna-se necessário existirem métodos de controlo capazes de identificar desvios do planeado. Sempre que necessário devem ser tomadas medidas corretivas sobre o plano de maneira a adequa-lo as condições externas e ao desempenho interno.

As funções do planeamento e controlo da produção pode ser classificadas em:

- Função de Pré-planeamento é um planeamento de alto nível que define as estratégias baseadas em análises de mercado e previsões de procura. Tem a função de desenvolve novos produtos e processos, adquirir novas máquinas, estabelecer capacidades e introduzir modificações no *layout* do sistema produtivo. O Planeamento Estratégico e o Plano Industrial e Comercial são normalmente tratados nesta fase;

- Função de Planeamento é onde são dispostos os recursos produtivos (Máquinas, Materiais e Homens) e definidos os métodos de produção, de maneira a atingir os objectivos operacionais da organização. O Plano Diretor da Produção, o Calculo das Necessidades Liquidas e o Escalonamento, podem ser aqui inseridos, nesta fase;

- Função de Controlo deve ocorrer durante todo o processo de planeamento, através da inspeção dos produtos acabados e análise das operações em curso. Um bom sistema de comunicação, entre os diversos intervenientes do processo produtivo, permite implementar medidas corretivas que melhor adeqúem o planeado à realidade [4].

2.6. Classificação dos sistemas de produção

Embora o propósito de um sistema produtivo, transformar *inputs* em *outputs*, seja comum a todas as unidades de produção, a especificidade de cada organização não permite uma abordagem completamente abrangente para as analisar todas do mesmo modo. De maneira a enquadrar cada organização e definir os métodos a serem implementados, as várias características das organizações devem ser estudadas e classificadas.

A classificação de uma unidade produtiva não e consensual; existem muitos factores que diferenciam as várias organizações [1] [5].

2.6.1. Os quatro V's

Existe uma clara distinção entre as organizações, baseada em quatro pontos que podem ser usados como métodos de classificação [3]:

- **Volume das séries de fabrico:** O volume das séries de fabrico é uma característica essencial para a classificação de uma unidade produtiva. Será possível dizer que uma organização que produz grandes séries deverá ser gerida segundo os mesmos princípios que uma organização que produz séries unitárias? De facto, a classificação de um sistema produtivo segundo o volume das suas séries permite retirar uma série de ilações quanto ao seu funcionamento e a sua relações com o mercado. Uma unidade que produz grandes séries, habitualmente fá-lo para o armazém, enquanto uma unidade que produz em séries unitárias o faz por encomenda. Da mesma maneira, é possível verificar uma clara relação entre volume das séries e a gama operatória;

- **Variabilidade de *outputs*:** A variabilidade de *outputs*, representa a gama de produtos que saem do sistema. Mais uma vez, será possível dizer que uma unidade produtiva que produz um único produto, deverá ser analisada da mesma maneira que uma unidade que produz uma vasta gama de produtos? A variabilidade de *outputs* não é totalmente independente do volume das séries a produzir. É normal que a variabilidade

e o volume sejam inversamente proporcionais, isto é, quanto menor for o tamanho das séries, maior variabilidade de produtos deve existir;

- **Variação de procura:** A variação da procura, identifica como os *outputs* da unidade produtiva são absorvidos pelo mercado, isto é, classifica a estabilidade da procura. Um sistema produtivo que tem um consumo estável, estará em melhores condições para produzir para armazém, com a segurança de que os seus *outputs* acabarão por ser absorvidos pelo mercado, por sua vez, uma unidade produtiva que tem uma procura muito incerta terá naturalmente a tendência para apenas produzir por encomenda;

- **Visibilidade do cliente:** Refere-se a visibilidade que os cliente têm acerca do processo produtivo, isto é, define o acesso que o cliente tem ao processo de transformação. Tipicamente, uma unidade de serviços permite muito maior visibilidade ao cliente, que está muitas vezes presente durante o processo; no entanto na produção de produtos, a visibilidade é muito diferente entre sistemas produtivos. Geralmente, unidades que produzem em séries muito grandes, apresentam um menor grau de acesso ao cliente, e a produção unitária, habitualmente, garante um relação de proximidade com cliente durante o processo de transformação.

Este tipo de classificação de um sistema produtivo não é exaustiva, pois existem muitos outros factores passíveis de permitir um diferenciação das organizações. No entanto a inter-relação entre os diferente pontos é clara e pensa-se que estes podem ser suficientes para retirar as ilações necessárias para um correta análise de um sistema produtivo [1] [3].

2.6.2. Classificação por implantação

A classificação por implementação de um sistema produtivo, parece ser a mais utilizada nos problemas de escalonamento.

Afirmar que um sistema produtivo funciona com uma dada implementação, é normalmente suficiente para deduzir o resto das suas características. Vão ser considerados três tipos de implementação: Linha de Fabrico (*Flow Shop*); Célula de Fabrico; Oficina de Fabrico (*Job Shop*) [1] [2] [5].

2.6.2.1. Linha de fabrico

É tipo de implantação mais comum quando se trata de uma unidade produtiva a transformar grandes séries com pouca variabilidade. Esta implantação caracteriza-se pela produção que percorre sempre o mesmo percurso ao longo da unidade. Os equipamentos de produção são altamente especializados, sendo implementados com a quase exclusiva finalidade de produzir uma baixa gama de produtos. Este tipo de produção utiliza, habitualmente, máquinas autónomas.

O facto de os recursos serem exclusivamente capazes de produzir uma baixa gama de produtos não permite uma grande flexibilidade, ao mesmo tempo as operações em série exigem a existência de uma manutenção preventiva, pois a falha de qualquer máquina leva a paragem da linha de fabrico.

Na implementação em linha de fabrico a movimentação entre postos de trabalho é, habitualmente, automática e a produção é feita para armazém. A implantação em linha de produção garante um baixo custo e uma elevada qualidade dos produtos. Na tabela 3 é possível analisar as principais vantagens e desvantagens características deste tipo de implantação [1] [2] [5].

Tabela 3 - Linha de Fabrico

Vantagens	Desvantagens
Tempo de ciclo reduzido	Manutenção preventiva
Taxa de utilização elevada	Pouca flexibilidade
Poucos produtos em curso	Investimento elevado
Baixo custo por unidade produzida	Dependência da operação mais lenta

2.6.2.2. Célula de fabrico

A célula de fabrico é, habitualmente, utilizada quando o sistema produtivo combina um razoável número de produtos em séries médias; isto deve-se ao facto de um único produto não ter consumo suficiente para justificar a afectação de recursos em exclusividade. A disposição física dos recursos utilizados é tipicamente em série, pois o procedimento habitual é a produção de um único produto em cada intervalo de tempo.

Assim, as células de fabrico são vocacionadas para a produção de famílias de produtos. A capacidade de executar mais do que uma produto, introduz um maior grau de flexibilidade a este tipo de implantação quando comparada com a linha de fabrico. O facto da implantação ser em série leva, tal como na implantação por linha de fabrico, à necessidade de recorrer à manutenção preventiva, pois uma falha numa única máquina pode levar à suspensão da produção durante o período de reparação.

O facto de os recursos terem que alternar entre diferentes tarefas, levou ao aparecimento de tempos não produtivos para preparação das máquinas; estes tempos de preparação (*Setup*) devem ser o mais curtos possível, para aumentar a rendibilidade do sistema. Com o aparecimento de novas filosofias de produção que dão especial importância à flexibilidade do sistema produtivo, várias técnicas para a diminuição dos tempos de preparação foram desenvolvidas, como o SMED (*Single-Minute Exchange of Die*) e posteriormente o OTED (*One-Touch Exchange of Die*). Na implantação em célula de fabrico, a produção é normalmente feita para armazém.

Na tabela 4 é possível analisar as principais vantagens e desvantagens características deste tipo de implantação [1] [2].

Tabela 4 - Célula de Fabrico

Vantagens	Desvantagens
Taxa de utilização razoável	Fluxo irregular
Custo médio por unidade	Complexidade de planeamento
Flexibilidade	Existência de tempos de preparação

2.6.2.3. Oficinas de fabrico

Usadas normalmente para produção descontínua em pequena séries, são habitualmente usadas para produtos personalizados segundo as especificações do cliente. Este tipo de implantação é utilizado quando a variabilidade de produtos é grande, necessitando de um conjunto de recursos capazes de executar um variado tipo de operações.

A organização das máquinas é feita em oficinas funcionais, onde se encontram vários equipamentos com funções semelhantes.

Cada produto, devido às suas especificações, vai necessitar de um número de operações numa dada sequência, o que irá determinar o seu percurso pelo sistema produtivo. O facto de dois produtos poderem ter rotas completamente distintas vai dificultar a coordenação da unidade, sendo, habitualmente, de enorme complexidade.

Contrariamente à implantação em linha de fabrico, ou mesmo em célula de fabrico, o número de unidades em curso de fabrico é elevado, tornando-se uma parte significativa do inventário da organização. A flexibilidade é a principal vantagem deste tipo de implantação, e a coordenação e diminuição dos tempos de preparação um dos maiores desafios.

Na tabela 5 é possível analisar as principais vantagens e desvantagens características deste tipo de implantação [1] [2] [5].

Tabela 5 - Oficina de Fabrico

Vantagens	Desvantagens
Alta flexibilidade	Custo alto por unidade produzida
Baixo custo de Implementação	Muito produtos em curso de fabrico
Fomenta a criatividade e inovação	Complexidade de planeamento

2.6.3. Implantação e os quatro V's

A classificação de um sistema produtivo pela sua implantação permite retirar uma série de ilações acerca do volume das suas séries, da variabilidade dos seus produtos, da variância da procura e da visibilidade por parte dos clientes.

Assim, uma empresa que disponha os seus recursos produtivos em linha, habitualmente, tem pouca variabilidade dos seus *output* que são produzidos em séries muito volumosas. O cliente tem pouca visibilidade do processo e a variação da procura também é reduzida. Um empresa que disponha dos seus recursos produtivos em células de fabrico,, habitualmente, tem média variabilidade de *outputs*, produz em volumes médios e existe variação na procura. Por fim, uma empresa que disponha os seus recursos produtivos em oficina de fabrico, tem muita variabilidade de *outputs*, produz séries pouco volumosas, existe maior incerteza na procura e o cliente tem maior acesso ao processo de produção.

A figura 4 permite analisar essas mesmas relações. Torna-se importante notar que essas são as relações habituais, e não devem ser consideradas sempre verdadeiras [1].

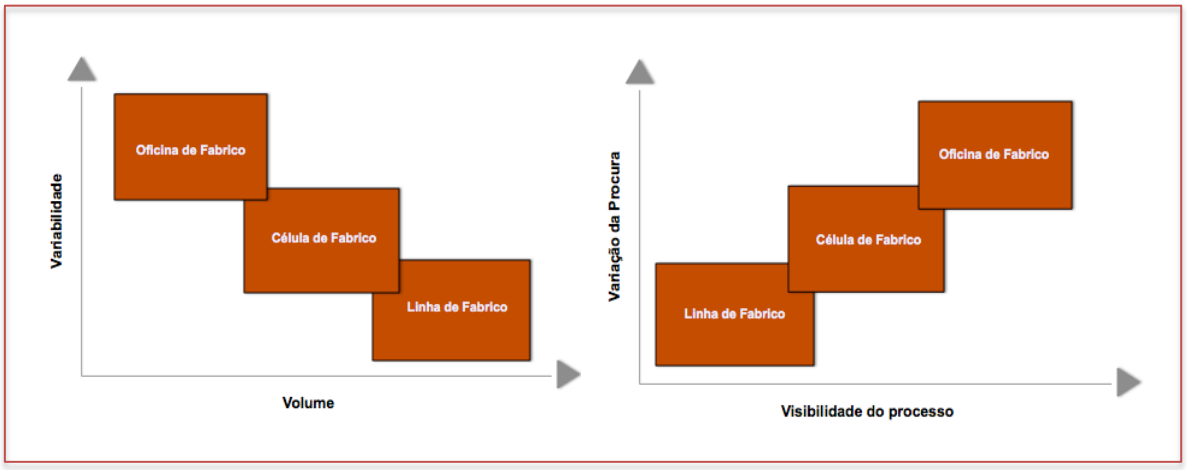

Figura 4 - Implantação e os Quatro V's [1]

2.7. Função do escalonamento numa empresa

O escalonamento da produção desempenha um papel vital nas organizações. Este pode ser definido como o processo de decisão que procura fazer uso eficiente dos recursos de produção, e assegurar a rápida execução dos trabalhos por forma a fazer a entrega do produto no prazo estipulado. Esta função esteve sempre presente nas organizações, mas nunca foi tão preponderante como atualmente. Os problemas com que as organizações se deparam são cada vez mais complexos, devido ao grande número de tarefas que têm de ser realizadas simultaneamente, com períodos de produção cada vez menores.

Em contexto industrial o escalonamento das operações surge dentro de uma hierarquia de decisões, precedido por decisões de natureza mais simples. O escalonamento pode ser inserido na função de planeamento que irá determinar os produtos, os recursos de produção disponíveis e os volumes a produzir num determinado intervalo de tempo. Depois de conhecidos os recursos disponíveis, os volumes e prazos de produção, é o escalonamento que deverá determinar como as operações vão ser executadas: que recursos vão ser alocados a cada operação e qual é a sequência de processamento das diferentes tarefas.

Um bom escalonamento traduz-se no aumento da capacidade competitiva da empresa através do aumento da eficiência na utilização dos seus recursos. Pode procura diminuir os tempo de processamento, reduzir os atrasos ou rentabilizar a utilização dos recursos.

O escalonamento tem um papel fundamental não apenas em unidades produtivas, como em sistemas de processamento informático, onde os recursos produtivos são substituídos por capacidade computacional. Assim é possível falar em escalonamento em contextos produtivo, de transporte, de distribuição ou de capacidade computacional [9] [10].

2.8. Conclusão

Neste capítulo foram apresentadas de uma forma sucinta as diferentes fases de planeamento em sistemas de produção. Entre essas fases, o escalonamento desempenha um papel importante pois refere-se à afectação dos recursos às diferentes operações. Sem escalonar as tarefas de forma adequada, a capacidade instalada é desperdiçada, levando à ineficiência da organização.

Foram ainda descritos métodos de classificação de sistemas produtivos. A classificação é importante pois irá permitir definir como melhor abordar o problema de escalonamento. Problemas dentro do mesmo tipo de tipologia podem ser abordados de maneira semelhante. Assim, duas organizações a operar numa tipologia de Linha de Fabrico que pretendam minimizar o tempo máximo de atraso de uma encomenda podem abordar o problema de escalonamento de uma forma idêntica, independentemente das características do produto, do número de máquinas ou do número de tarefas a executar.

3. PROBLEMA DE ESCALONAMENTO

Neste capítulo, será descrito, em maior detalhe, o problema do escalonamento. Para melhor se compreender a função do escalonamento vão ser descritos os vários elementos de um problema de escalonamento, os objectivos, as medidas de desempenho habitualmente usadas e apresentados vários métodos para resolução dos problemas de escalonamento clássicos. Finalmente será definida a fase da afectação do problema de escalonamento.

3.1. Introdução

Com a Revolução Industrial o processo produtivo sofreu alterações significativas. A produção artesanal e pouco especializada foi substituída por empresas centralizadas, com estruturas organizacionais detalhadas. Essas fábricas eram bastante simples, produzido um reduzido número de produtos, em grandes séries. O aumento da produtividade era feito através da redução dos tempos de preparação, tentando rentabilizar os equipamentos caros, aumentado a sua taxa de utilização. A coordenação das atividades era da responsabilidade de um chefe de oficina que comprava as matérias primas, geria a produção e despachava os produtos. O planeamento da produção era informal, resumindo-se à lista que descrevia quando cada tarefa devia começar a ser produzida e qual era a sua data de entrega.

Por volta de 1890, houve um aumento exponencial na complexidade das empresas. O número de produtos aumentou rapidamente, o que não permitia aos chefes de oficina continuar a controlar toda a produção. Frederick Taylor, propõe a criação de um departamento de planeamento, para gerir o nível de inventário e monitorizar as operações. O departamento de planeamento calculava as necessidades de produção, lançava as ordens e escalonava as operações. No escalonamento da produção não participava apenas o departamento de planeamento, participava também do chefe de oficina, que conhecia o processo produtivo detalhadamente [8].

No inicio do século XX, Henry Gantt inventou um método de controlar o escalonamento da produção, através de diagramas. Os diagramas de Gantt, permitem mostrar graficamente as relações entre o desempenho planeado e o desempenho real. Nos seus diagramas, Gantt mede as tarefas através do tempo necessário para as completar e representa essa medida graficamente, através do espaço que ocupam no diagrama. A representação visual das ordens lançadas permitiu tornar mais intuitivo o planeamento da produção, [8].

Embora os diagrama de Gantt se mantenham como uma das ferramentas mais habituais no planeamento e controlo da produção, com o desenvolvimento e introdução dos computadores pessoais o método manual foi substituído por aplicações informáticas, capazes de produzir o plano de produção com maior detalhe em menor tempo. O escalonamento da produção também sofreu modificações profundas com o desenvolvimento das tecnologias de informação. Computadores com maior capacidade de processamento são capazes de resolver algoritmos complexos mais rapidamente, o que permitiu desenvolver métodos de escalonamento mais eficientes. Durante as últimas décadas, o desenvolvimento de métodos e ferramentas para o escalonamento da produção, tem sido significativo, [8]:

- Anos 1940s: Programação linear; Simplex;
- Anos 1950s: Algoritmo de Johnson; Regra de prioridade EDD e STP;
- Anos 1960/70s: *Branch-and-Bound*; Programação por restrições;
- Anos 1980/90s: Heurísticas de pesquisa local; Heurísticas evolutivas.

Atualmente o desenvolvimento de novos métodos e ferramentas para o escalonamento da produção tem acompanhado os desenvolvimentos nas ciências computacionais [8].

3.2. Afectação e Sequenciação/Calendarização

O problema de escalonamento representa um número de tarefas que tem que ser executadas com certos recursos. Para resolver um problema de escalonamento é necessário conhecer as tarefas detalhadamente, bem como os recursos disponíveis. Sem conhecer as características das tarefas e as capacidades dos recursos, é impossível escalonar as tarefas corretamente. Conhecendo as tarefas e os recursos disponíveis, é possível determinar a maneira como essas tarefas deverão ser distribuídas pelas recursos e qual deverá ser a sua sequência e calendário para o seu processamento. Assim, um problema de escalonamento procura responder às seguintes questões:

- Dado um conjunto de atividades e um conjunto de recursos, qual a melhor forma de distribuir os recursos pelas atividades de modo a maximizar o desempenho?
- Qual será a sequência de atividades, que melhor se adequa aos objectivos económicos e operacionais ?

Um problema de escalonamento pode ser dividido em problemas de afectação e problemas de sequenciação/calendarização, embora existam casos que apenas tratam de um tipo de problemas. Os problemas de afectação, determinam que recursos serão atribuídas a que tarefas. É um problema de decisão, pois em ambientes onde existem recursos em paralelo, é necessário escolher o recurso que vai executar determinada tarefa. Os problemas de sequenciação/calendarização, determinam em que sequência deverão ser executas as tarefas, isto é, por que ordem serão executas as tarefas em cada um dos recursos.

Existe uma forte relação entre as fases de afectação e sequenciação/calendarização, sendo muitas vezes difícil fazer uma clara separação entre ambas. Em problemas onde todos os recursos estão dispostos de forma paralela, as decisões de afectação são tomadas antes de ser definida uma sequência/calendário de execução para as tarefas. Em problemas onde os recursos estão dispostos de formas mais complexas, a separação entre ambas as fases do escalonamento é mais difusa.

Apesar da aparente simplicidade da afectação e sequenciação/calendarização, existem factores que dificultam essas decisões. Na realidade, parte dos problemas de escalonamento estão acompanhados por restrições que tornam a sua resolução intratável por métodos exatos [6] [8] [10].

3.3. Elementos do problema de escalonamento

A teoria do escalonamento foi desenvolvida para resolver problemas em unidades produtivas, pelo que, para caracterizar os problemas foi utilizada uma terminologia similar à utilizada numa fábrica. Os recursos são habitualmente chamados de máquinas, as atividades são chamadas de tarefas e em problemas onde as tarefas são executadas em várias máquinas, estas podem ser divididas em operações. Por fim, o ambiente do problema de escalonamento é, habitualmente, designado por oficina.

Assim para caracterizar um problema de escalonamento, é necessário conhecer as características dos seus elementos, como:

- **Máquina** – É um recurso técnico ou humano necessário para a execução de uma tarefa, disponível em quantidade limitada, com uma determinada capacidade. O recurso é dito renovável se ficar novamente disponível uma vez concluída a tarefa. Se o recurso não for renovável é um recurso consumível. Os recursos serão disjuntivos, se apenas poderem ser utilizados numa tarefa de cada vez, ou cumulativos, se poderem ser partilhados por varias tarefas simultaneamente, respeitando a sua capacidade.

- **Tarefa** – É uma atividade de trabalho localizada no tempo, onde é conhecido o momento onde pode começar a ser executada e o fim da sua execução. É caracterizada pela sua duração e pelos recursos necessários para ser executada. Pode ser considerado que a intensidade no consumo de recursos é constante durante a sua execução.

- **Operação** – É uma parte de uma tarefa que apenas utiliza uma única máquina, pode ser uma operação de preparação ou uma operação de produção. Assim, qualquer tarefa que necessite de ser executada por várias máquinas, antes da sua conclusão, pode ser dividida em operações onde apenas utilizam um recurso.

- **Oficina** – Local onde estão dispostas as máquinas afectas à produção. As máquinas podem estar dispostas em linha de fabrico, célula de fabrico ou oficina de fabrico. A maneira como as máquinas estão dispostas vai condicionar a circulação das tarefas na oficina, com todas as operações a sucederem em rotas pré-definidas.

Uma vez conhecidas as características das máquinas, tarefas e operações, é possível melhor definir o problema de escalonamento [7] [10].

3.4. Objectivos e modelos

É muitas vezes complicado medir o desempenho das decisões de escalonamento. Os custos associados à forma como as operações vão ser executadas, são difíceis de identificar ou quantificar. Assim os três objectivos mais comuns num problema de escalonamento são:

- Tempo de execução;
- Pontualidade;
- Produtividade.

O tempo de execução, mede o tempo necessário para processar uma tarefa. A pontualidade, é a diferença entre o fim da execução de uma tarefa e o sua prazo de conclusão esperado, isto é, a capacidade de uma tarefa cumprir o sua data de entrega. A produtividade, mede a quantidade de trabalho concluído durante um determinado período de tempo. Enquanto o tempo de execução e a pontualidade medem o desempenho de uma única tarefa, a produtividade pode ser utilizada para medir o desempenho de uma série de tarefas [10].

Podem existir outros tipos de objectivos, que são muito mais difíceis de quantificar. Custos associados à imagem da organização, ou custos referentes ao descontentamento dos funcionários, não são fáceis de quantificar ou de representar numa expressão matemática.

Para caracterizar os principais modelos de escalonamento, são usadas as especificidades das tarefas e a forma como os recursos estão dispostos. Em modelos onde apenas exista uma máquina, é natural que as tarefas apenas tenham uma única operação, enquanto que em modelos onde existam várias máquinas é normal que as tarefas possam ser divididas em várias operações distintas.

Os modelos onde as tarefas a escalonar não mudam com o passar do tempo, são chamados de modelos estáticos; os modelos onde novas tarefas podem surgir são chamados de modelos dinâmicos. Embora os modelos dinâmicos possam parecer mais próximos da realidade, os modelos estáticos, mais simples, podem ser utilizados para simplificar os modelos dinâmicos, mais complexos [10].

Em modelos onde as condições são assumidas como constantes, são chamados de modelos determinísticos, enquanto aqueles onde existe incerteza, com uma probabilidade associada, podem ser designados por modelos estocásticos [10].

3.5. Modelos teóricos vs. Problemas reais

Embora o problema do escalonamento tenha sido alvo de um amplo estudo nas últimas décadas, os problemas estudados são tendencialmente focados em condições muito particulares, que na realidade nem sempre se verificam. Habitualmente os problemas de escalonamento teóricos, diferem dos problemas reais nos seguintes pontos:

- Normalmente os problema de escalonamento teóricos, estudam sistemas estáticos com características constantes e conhecidas. Tendem a focar-se em problemas onde são escalonadas *n* tarefas, o que na realidade raramente acontece, pois à medida que as tarefas são concluídas, novas ordens são lançadas. É dada pouca atenção à reordenação das tarefas, o que face à incerteza associada aos problemas reais, acontece regularmente. O peso das tarefas também é habitualmente constante, o que na realidade nem sempre é verdade. Da mesma maneira, não costumam ser incluídas questões de preferência nos recursos utilizados [6].

- Os problema teóricos preocupam-se com a optimização de um único objectivo, tal habitualmente não acontece em problemas reais. Na realidade o escalonamento da produção, procura soluções capazes de aumentar a eficiência da organização considerando diversos critérios de otimização. Recentemente, os problemas multiobjectivo tem sido alvo de múltiplos estudos, com o propósito de aproximar o estudo teórico do escalonamento, da realidade nas empresas [6].

- Os problemas de escalonamento teóricos, assumem características mais simples que aqueles que se verificam na realidade. A disposição das máquinas é habitualmente mais simples do que em muitos casos reais, onde as máquinas estão muitas vezes dispostas de uma forma muito complexa. Da mesma maneira, não se costumam assumir restrições de disponibilidade das máquinas, o que na realidade acontece frequentemente. Por fim, os modelos teóricos não têm em consideração factores como a motivação ou a aprendizagem [6].

Mesmo tendo em consideração as diferenças entre os modelos teóricos e os modelos reais, o estudo realizado nas ultimas décadas tem sido extremamente útil. Os modelos teóricos tem permitido desenvolver modelos matemáticos, para um elevado número de problemas com que as empresas se deparam [6].

3.6. Representação

Com o estudo do problema de escalonamento da produção, a variedade de modelos cresceu rapidamente. Esse crescimento levou à necessidade de desenvolver sistemas de representação, que fossem capazes de descrever de forma sucinta o modelo que representava. Embora os sistemas de representação não caracterizem todos os problemas, permitem caracterizar os principais problemas de escalonamento.

Existem atualmente dois métodos para a representação de escalonamento, um descrito por Conway, com quatro campos, mais apropriado para problemas menos complexos e um descrito por Graham e Pinedo, com três campos, que permite representar problemas mais complexos. Durante o trabalho de mestrado vai ser utilizada o modelo de representação de Graham, que permite descrever o problema de escalonamento, mais detalhadamente [12].

O número de tarefas é representado por *n*, e o número de máquinas representado por *m*. Com *j*, a identificar a tarefa e o *i*, a identificar a máquina e sendo o número de máquinas (*n*) e o número de tarefas (*m*) sempre finito, é possível definir:

- **Tempo de Processamento** (p_{ij}) - Representa o tempo de processamento da tarefa *j*, na máquina *i*. Se *i*, não existir é porque o tempo de processamento da tarefa *j*, não depende da máquina onde vai ser processada ou a tarefa vai ser processada apenas numa única máquina.

- **Data de Entrega** (d_j) - Representa a data de entrega da tarefa *j*, que ao não ser cumprida leva a organização a incorrer numa, possível, penalidade. A data de entrega indica uma data acordada com um cliente, que se não cumprida representa um custo para a organização.

- **Peso** (w_j) – Representa a importância da tarefa *j*, quando comparado com as outras tarefas do sistema. Uma tarefa com um peso superior deverá ter prioridade sobre tarefas com pesos inferiores. Se não existirem pesos é porque todas as tarefas tem uma prioridade semelhante.

Um problema de escalonamento vai ser representado por $\alpha \mid \beta \mid \gamma$. Onde α, descreve como os recursos, afectos a produção, estão dispostos, β descreve as restrições do processo e γ descreve o critério de optimização [6] [9] [12].

3.6.1. Tipo de implantação

Para α, são possíveis as seguintes alternativas:

- **Máquina Única** (1) - Para o caso de existir uma única máquina onde todas as tarefas deverão ser processadas. É o tipo de implantação mais simples e por isso é muito utilizado para resolver sub-problemas de implementações mais complexas.

- **Máquinas Idênticas em Paralelo** (*Pm*) - Quando existem *m* máquinas idênticas, em paralelo. Representa o caso da tarefa *j* apenas necessitar de ser processada numa das *m* máquinas, todas elas com características idênticas.

- **Máquinas com Velocidades Diferentes em Paralelo** (*Qm*) - Quando existem *m* máquinas com velocidades diferentes, em paralelo. Representa o caso da tarefa *j* apenas necessitar de ser processada numa das *m* máquinas, com uma velocidade v_i.

- **Máquinas Diferentes em Paralelo** (*Rm*) - Quando existem *m* máquinas com velocidades dependentes da tarefa que é afecta. Representa o caso da tarefa *j* apenas necessitar de ser processada numa das *m* máquinas, com uma velocidade v_{ij}.

- **Linha de Fabrico** (*Fm*) - No caso de existirem *m* máquinas em série. Cada tarefa necessita de ser processada em cada uma das *m* máquinas na mesma sequência. Assim uma tarefa após ser processada na máquina *m*, vai transitar para máquina *m+1*.

- **Linha de Fabrico Flexível** (*FF*c) - No caso de existirem *c* centros, cada um com *m* máquinas idênticas. Cada tarefa necessita de ser processada em cada um dos níveis por uma das *m* máquinas disponíveis.

- **Oficina de Fabrico** (*Jm*) - Para o caso de existirem *m* máquinas e em que cada tarefa percorre uma rota especifica através do sistema produtivo. Existem oficinas de fabrico onde cada tarefa apenas visita cada máquina uma única vez e aquelas onde cada tarefa pode visitar a mesma máquina mais que uma vez.

- **Oficina de Fabrico Flexível** (*FJc*) - Para o caso de existirem *c* centros de trabalho, cada um com *m* máquinas idênticas. Cada tarefa necessita de ser processada em cada um dos centros de trabalho. Existem oficinas de fabrico onde cada tarefa apenas visita cada centro de trabalho uma única vez .

- **Oficina Aberta** (*Om*) - Para o caso de existirem *m* máquinas e cada tarefa ter que ser processada por cada uma, no entanto o tempo de processamento de certas tarefas em algumas máquinas pode ser zero.

Assim, para preencher o campo α, é necessário conhecer a disposição dos recursos produtivos. Este tipo de classificação é feita analisando a oficina e posteriormente vai ser analisada mais detalhadamente [6] [9].

3.6.2. Restrições do processo

Para β são possíveis as seguintes alternativas:

- **Data de Lançamento** (r_j) - Representa a impossibilidade da tarefa *j* começar a ser processada antes da data r_j. Se não aparecer em β, é porque todas as tarefas podem começar a ser executadas imediatamente.

- **Interrupções** (*prmp*) - Representa a possibilidade do processamento duma tarefa poder ser interrompido antes desta estar concluída. A quantidade de trabalho na tarefa que é interrompida não é perdido, o que permite que uma tarefa cujo processamento foi interrompido possa ser concluída posteriormente.

- **Relações de Precedência** (*prec*) - Representa a existência de relações entre a execução das diferentes tarefas. Uma tarefa que tenha uma relação de precedência sobre outra, tem que ser processada antes dela.

- **Tempos de *Setup* dependentes da Sequência** (s_{ik}) - Representa a existência de tempos de preparação, dependentes da sequência em que as tarefas são processadas. Assim s_{ik}, representa o tempo de preparação necessário entre as tarefa *j* e *k*.

- **Família de Tarefas** (*fmls*) - Representa a existência de famílias de tarefas. Dentro de cada família, as tarefas podem ter tempos de processamento diferentes mas não necessitam de tempo de preparação entre elas.

- **Produção por Lote** (*batch(b)*) - Representa a possibilidade de uma máquina executar mais que uma tarefa simultaneamente. Assim a máquina pode executar *b* tarefas simultaneamente, que apenas serão concluídas quando a tarefa com o tempo de processamento mais longo for concluída.

- **Avarias** (*brkdwn*) - Representa a disponibilidade das máquinas, considerando a manutenção preventiva e a manutenção não planeada. Se as máquinas estiverem sempre disponíveis esta opção não deverá aparecer em β.

- **Permutações** (*prmu*) - Representa a impossibilidade de alterar a sequência de tarefas após execução na primeira máquina. Este tipo de restrição apenas tem sentido para unidades produtivas a operar como linha de fabrico.

- **Sem Espera** (*nwt*) - Representa a necessidade de uma tarefa não esperar entre as diferentes operações de uma Linha da Fabrico. Isto implica que uma tarefa apenas pode começar a ser executada quanto estiverem reunidas as condições para a tarefa poder circular através de toda a linha sem qualquer tempo de espera.

- **Recirculação** (*rcrc*) - Representa a possibilidade de uma tarefa poder visitar uma máquina mais do que uma vez. Esta característica apenas tem sentido em Oficinas de Fabrico ou Oficinas de Fabrico Flexíveis.

Assim, para preencher o campo β, é necessário conhecer as especificidades do problema de escalonamento que irá ser tratado. Ao contrário do campo α, o campo β pode conter mais do que uma entrada, se o problema estiver sujeito a várias restrições; da mesma maneira o campo β pode encontrar-se vazio, se o problema não apresentar nenhuma das características descritas [6] [9].

3.6.3. Funções objectivo

Para γ são possíveis as seguintes alternativas:

- ***Makespan*** (C_{max}) – Representa o tempo desde do lançamento de ordem até à sua conclusão. Vai determinar o tempo que todas as tarefas demoram a ser processadas. Para minimizar o *Makespan*, é necessário optimizar a utilização dos recursos.

- **Atraso Máximo** (L_{max}) - Define a pior violação às datas de entrega. É definida pela tarefa que tenha a pior relação entre sua data de conclusão e a sua data de entrega.

- **Soma Pesada dos Atrasos Positivos** ($\sum w_j T_j$) – Define a soma ponderada, das tarefas que não vão cumprir as suas datas de entrega. Isto é, tarefas em que a data de conclusão é maior que a date de entrega.

- **Número Ponderado de Tarefas Atrasadas** ($\sum w_j U_j$) - Define uma soma ponderada de todos os trabalhos que não vão cumprir as suas datas de entrega. Ao contrário do Atraso, *U* representa as tarefas que não vão cumprir a sua data de entrega, por uma variável binária.

Assim, para preencher o campo γ é necessário conhecer o objectivo do problema de escalonamento. É importante notar que existem mais funções objectivo do que aquelas apresentadas. Posteriormente vai ser apresentada uma lista mais detalhada com as medidas de desempenho num problema de escalonamento [6] [9].

3.7. Classes do escalonamento

Os planos de escalonamento da produção podem ser classificados:

- **"Semi-Ativo"**, quando para uma dada sequência de processamento em cada máquina, cada operação é iniciada o mais cedo possível. Neste plano não e possível começar uma operação mais cedo sem modificar a ordem das tarefas numa das máquina [12].
- **"Ativo"**, é uma subclasse dos planos "Semi-Ativos". Neste plano, nenhuma operação pode ser iniciada mais cedo sem provocar um atraso noutra operação [12].
- **"Sem Atrasos"** é uma subclasse dos planos "Ativos". Neste plano nenhuma máquina fica inativa quando pode iniciar o processamento de uma operação, isto é, só existem tempos mortos se não houver qualquer operação pronta para ser processada [12].

Um escalonamento ótimo é sempre "Ativo", é uma escalonamento "Ativo" pode ser também "Sem Atrasos", como é possível ver na figura 5. Nem sempre um escalonamento ótimo é "Sem Atrasos", no entanto para problemas que permitam interrupções, o escalonamento ótimo é sempre "Sem Atrasos" [6] [12].

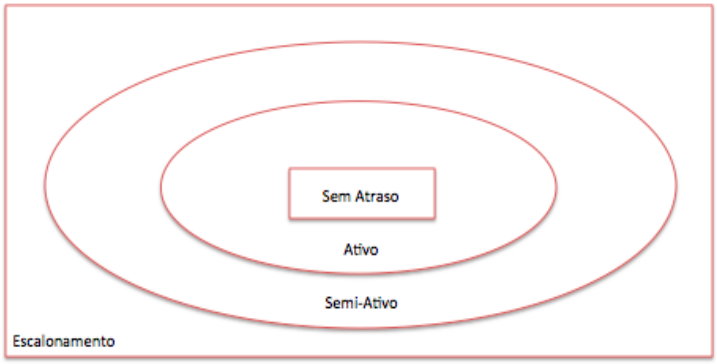

Figura 5 – Classes do Escalonamento [6]

3.8. Complexidade computacional

Em termos de dificuldade de resolução, os problemas de escalonamento são classificados de acordo com a teoria de complexidade computacional. O esforço necessário para resolver um algoritmo é habitualmente uma função dependente da dimensão do problema (*n*), chamada de função de complexidade. Para funções de complexidade de natureza polinomial, os algoritmos são designados por algoritmos polinomiais ou de classe P, quando a função de complexidade não pode ser descrita por uma função polinomial, então são designados por algoritmos não-polinomiais, ou de classe NP. Os problemas de escalonamento para os quais não são conhecidos algoritmos polinomiais, são de difícil resolução, devido a sua complexidade exponencial [12].

Por exemplo, dois modelos matemáticos com funções de complexidade k (Polinomial) e 3^k (Não-Polinomial), necessitam de um esforço computacional muito diferente. Se $k = 60$ e considerando cada operação de computação a demorar 1 μs, o primeiro algoritmo pode ser resolvido em 60 μs, enquanto o segundo irá demorar 1.3×10^{13} séculos. Essa disparidade aumenta com o aumento de *k*, na razão da função de complexidade.

No entanto, nem todos os problemas de escalonamento têm as mesmas características. Entre os problemas de pesquisa existem duas subclasses de problemas:

- Problemas de Optimização, problemas que procuram a melhor solução para um problemas sob determinadas condição [9];
- Problemas de Decisão, problemas que devem determinar se uma solução é ou não aceite como solução do problema [9].

Entre os problemas de escalonamento para os quais não são conhecidos algoritmos polinomiais, existem duas classes particulares. Os problemas NP-*complete*, que incluem muitos problemas de decisão de natureza combinatória, são equivalentes entre si. Isto é, se for encontrado um algoritmo polinomial capaz de resolver um destes problemas, então todos os outros problemas de decisão NP-*complete*, também podem ser resolvidos através algoritmos polinomiais, [9]. Os problemas NP-*hard*, são problemas de otimização em que o problema de decisão equivalente é NP-*complete*. Uma vez que os problemas de optimização e os problemas de decisão são computacionalmente equivalentes, os problemas NP-*hard*, são tão difíceis como os problemas NP-*complete* [9].

Os problemas NP-*hard* são por isso problemas de otimização para os quais não são conhecidos algoritmos eficientes de resolução. Um algoritmo é considerado eficiente se a sua complexidade temporal crescer de forma polinomial e não de forma exponencial com a dimensão do problema, isto é, permite a obtenção de soluções em tempo útil [12].

É possível ver um esquema onde são representados os problemas de natureza polinomial, não polinomial, os problemas NP-*complete* e os NP-*hard*, na figura 6 [9] [10] [12].

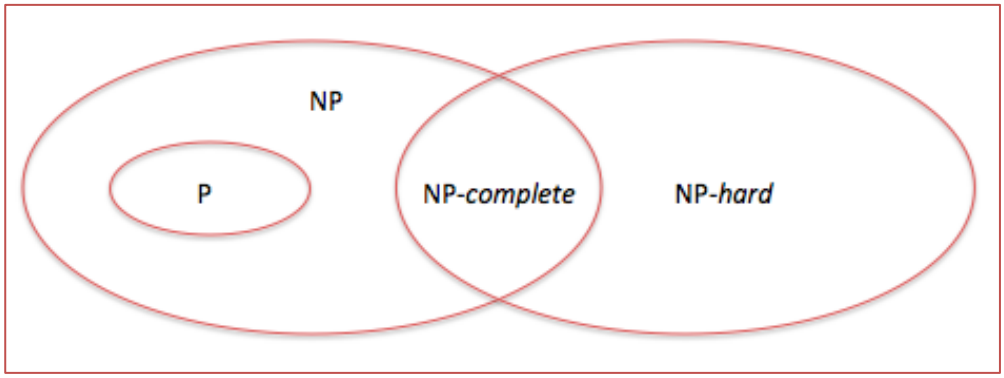

Figura 6 – Classes de Complexidade

3.9. Medidas de desempenho

O escalonamento vai determinar como uma empresa utiliza dos seus recursos. Define como as tarefas serão sequenciadas e decide que recursos lhes serão alocados. O plano de escalonamento pode ter objectivos diversos, com uma organização a dar prioridade à minimização do tempo de fabrico dos seus produtos, enquanto outra pode preferir minimizar o atraso das suas encomendas. Assim, é necessário definir métodos capazes de medir o desempenho do plano de escalonamento.

No escalonamento existem medidas de desempenho relacionadas com o tempo de execução, com as datas de entrega e com a utilização dos recursos disponíveis.

Para problemas onde o objectivo é diminuir o tempo de execução, existem as seguintes medidas de desempenho:

- **Data de Conclusão** (C_j) - Determina quando a tarefa *j* sai do sistema produtivo, isto é, define a data da conclusão da tarefa.

- **Tempo de Execução de uma Tarefa** (F_j) - Representa o tempo que a tarefa *j* demora a percorrer o sistema de produção, isto é, o tempo entre a data de lançamento e a data de conclusão. Pode ser calculado através da expressão:

$$F_j = C_j - r_j \quad (1)$$

- **Tempo Médio de Execução** ($\bar{F}$) - Representa o tempo médio que *n* tarefas demoram a percorrer o sistema de produção, isto é, a média da diferença entre data de conclusão de cada uma das tarefas e as suas datas de lançamento. Pode ser calculado por:

$$\bar{F} = \frac{1}{n}\sum_{j=1}^{n} F_j \quad (2)$$

- ***Makespan*** (C*max*) - Representa a data de conclusão da última tarefa a sair do sistema produtivo, isto é, última tarefa a estar concluída . Pode ser calculado pela expressão:

$$C_{max} = \max(C_j) \quad \forall\, j \in \{1 \dots n\} \quad (3)$$

Para problemas onde o objectivo é diminuir os atrasos, identificam-se as seguintes medidas de desempenho:

- **Atraso** (L_j) - Representa a relação entre o fim da produção das tarefas e as suas datas de entrega, isto é, a diferença entre as datas de conclusão e as datas de entrega. O atraso pode apresentar valores negativos se uma tarefa for concluída antes da sua data de entrega desejada. Pode ser calculada pela expressão:

$$L_j = C_j - d_j \quad (4)$$

- **Atraso Máximo** (L_{max}) - Representa a pior violação às datas de entrega, isto é, a tarefa que apresenta a maior diferença entre a data de conclusão e sua data de entrega. Pode ser calculada pela expressão:

$$L_{max} = \max|L_j| \quad \forall\, j \in \{1 \dots n\} \quad (5)$$

- **Atraso Positivo** (T_j) - Representa um atraso efetivo em relação à data de entrega. Pode ser calculado pela expressão:

$$T_j = \max(C_j - d_j, 0) = max(L_j, 0) \quad (7)$$

- **Atraso Negativo** (E_j) - Representa uma antecipação em relação às datas de entrega. Pode ser calculado pela expressão:

$$E_j = \max\big(0, -(C_j - d_j)\big) = max\big(0, -L_j\big) \qquad (8)$$

- **Atraso Positivo Médio** ($\bar{T}$) - Representa a média do atraso positivo entre todas as tarefas. Pode ser calculado pela expressão:

$$\bar{T} = \frac{1}{n}\sum_{j=1}^{n} T_j \qquad (9)$$

- **Tarefa em Atraso** (U_j) - Representa uma tarefa que não vai cumprir a data de entrega desejada. As tarefas que terminem antes da data de entrega não são consideradas, sendo apenas tido em conta o atraso positivo. Pode ser calculado pela expressão:

$$U_j = \begin{cases} 1 \; se \; C_j > d_j \\ 0 \; se \; C_j \leq d_j \end{cases} \qquad (10)$$

- **Numero de Tarefas em Atraso** (N_T) - Representa o número de tarefas em atraso, tendo apenas em consideração o atraso positivo. Pode ser calculado pela expressão:

$$N_T = \sum_{j=1}^{n} U_j \qquad (11)$$

- **Soma dos Custos de Posse e de Atraso** (*HT*) - Representa os custos em a empresa vai incorrer com o atraso. Uma vez que os custo de posse são habitualmente diferentes dos custo de atrasos, estes são representados por duas variáveis distintas. O custo de posse é representado por h_j e o custo de atraso por w_j. Pode ser calculado pela expressão:

$$HT = \sum_{j=1}^{n} \big(h_j \times E_j + w_j \times T_j\big) \qquad (12)$$

Para problemas onde o objectivo é aumentar a taxa de utilização, existem as seguintes medidas de desempenho:

- **Taxa de Utilização de uma Máquina** - Representa a utilização de uma máquina, isto é, a relação entre o tempo em que está a ser utilizada e a sua disponibilidade total. Pode ser calculado pela expressão:

$$\frac{1}{C_{max}} \sum_{j=1}^{n} p_{ij} \times 100 \qquad (13)$$

- **Taxa de Utilização de um Sistema** – Representa a utilização média de cada uma das máquinas do sistema produtivo e é normalmente dado em percentagem da disponibilidade total do sistema. Pode ser calculado pela expressão:

$$\frac{1}{C_{max} \times M} \sum_{i=1}^{M} \sum_{j=1}^{n} p_{ij} \times 100 \qquad (14)$$

O escalonamento da produção serve os principais objectivos da organização, sendo por isso difícil definir um único propósito. É fácil compreender que uma empresa poderá pretender diminuir o *makespan*, ao mesmo tempo que pretende eliminar atrasos e ainda maximizar as taxas de utilização dos seus equipamentos. Essa multiplicidade de objectivos pode mesmo tornar o problema impossível de resolver através de métodos de um único objectivo. Para resolver esse tipo de problemas é necessário recorrer a métodos multiobjectivo capazes de encontrar todas as soluções eficientes, num compromisso entre os diferentes parâmetros, definido pelo agente de decisão [6] [12].

3.10. Classificação dos problemas de escalonamento

Embora existam vários objectivos para o escalonamento, há características comuns a problemas em ambientes onde os recursos afectos à produção estão dispostos de forma semelhante. Assim, os problemas de escalonamento podem ser divididos em:

Problemas em Máquina Única - O escalonamento que trata de ambientes de uma única máquina é o mais simples e por isso mesmo o seu estudo tem particular importância. O seu estudo permite retirar ilações acerca de problemas em ambientes mais complexos. Muitas vezes problemas de escalonamento com tipologias mais complexas, são simplificados em vários sub-problemas de máquina única, que são mais fáceis de resolver. Problemas em ambientes fabris, onde é possível identificar um *bottleneck*, podem ser reduzidos a problemas de máquina única, muito facilmente.

Problemas em Máquinas Paralelas - Os problemas com máquinas em paralelo são tão importantes do ponto de vista teórico, pois são usados para abordar problemas em ambientes mais complexos, como do ponto de vista prático, pois muitas das organizações dispõem de recursos dispostos em paralelo.

Este tipo de problemas pode ser dividido em duas fases:

- Determinar que máquina vai executar cada tarefa.
- Determinar a sequência em que as tarefas vão ser executadas em cada máquina.

Depois de determinado quais são as máquinas que vão executar as diversas tarefas, é possível reduzir um problema em máquinas paralelas, em vários problemas de máquina única. Este tipo de problemas pode ser divido em problemas em que todas as máquinas tem a mesma velocidade, velocidades diferentes ou em problemas em que cada máquina tem características complemente diferentes.

Problemas em Linha de Fabrico - Descreve os ambientes onde as tarefas têm que ser executadas na mesma sequência de máquinas, ou seja, todas as tarefas devem percorrer o mesmo caminho através da fábrica. Normalmente as tarefas são empurradas através do sistema. No entanto, para isto poder acontecer tem que existir capacidade de armazenamento das tarefas em curso. Se não existir capacidade de armazenamento entre as máquinas, ou se o *buffer* de armazenamento estiver cheio, as tarefas na máquina antecedente terão que ficar em espera na máquina onde foram processadas.

É considerada uma linha de fabrico sequencial, aquela onde a sequência de tarefas não pode ser alterada entre as máquinas. Geralmente uma linha de fabrico sequencial é menos complexa que aquela onde a sequência pode variar entre as diferentes máquinas. Quando não existe armazenamento para as matérias em cursos, é possível assumir que a linha de fabrico é sequencial, pois as tarefas que são libertadas numa máquina têm que ser executadas imediatamente na máquina seguinte.

Deve também ser feita uma distinção entre as linhas de fabrico onde apenas existe uma máquina em cada nível e as linhas de fabrico onde existem varias máquinas em paralelo em cada nível. No último caso, as tarefas deslocam-se de nível para nível, onde podem ser executadas por várias máquinas diferentes, naquilo que é designado de “linha de fabrico flexível”.

Problemas em Oficina de Fabrico - Descreve ambientes de fabrico onde as tarefas têm que ser processadas em várias máquinas. Ao contrario da implantação em Linha de Fabrico, neste tipo de implantação as tarefas não necessitam de percorrer todas a oficina através da mesma rota.

Se uma tarefa puder visitar certas máquinas por mais que uma vez, então a implantação é em Oficina de Fabrico com Recirculação. Como é conhecida, à priori, a rota em que cada tarefa vai circular, é possível planear a maneira a melhor utilizar os recursos.

A implantação em Oficina de Fabrico é a mais difícil que tratar, devido ao número de rotas através do sistema produtivo. Isso levou ao desenvolvimento de várias heurísticas, que permitem encontrar soluções satisfatórias para os problemas em oficinas de fabrico.

Problemas em Oficina Aberta – Os problema de escalonamento em ambientes de oficina aberta são semelhantes aos problemas em oficina de fabrico com uma única diferença. Ao contrario dos problemas em oficina de fabrico, neste tipo de problemas as tarefas não têm uma rota fixa através do sistema. Isto é, não existem uma ordem pré-definida para as operações que constituem uma tarefa, o que torna os problemas em oficina aberta, problemas de natureza combinatória, não permitindo utilizar os mesmo métodos que são aplicados a problemas em oficina de fabrico ou linhas de fabrico.

Um caso particular dos problemas em oficina aberta, é quando estes permitem *preemptions*; nesse caso é possível resolver o problema muito facilmente, ao contrário do que acontece em problemas do mesmo tipo em oficinas de fabrico [6] [7].

3.11. Otimização combinatória

Um problema de escalonamento envolve decisões de como distribuir as tarefas no tempo e pelos recursos disponíveis, indicando quando, por quem e com o que as tarefas vão ser realizadas. Essas decisões envolvem uma escolha na sequência pela qual as tarefas devem ser executadas, o que se traduz num problema de natureza combinatória.

Um problema de natureza combinatória pode ser descrito como problema discreto, onde cada solução corresponde à ordenação para um número finito de tarefas. Cada solução pode ser caracterizada pela permutação na sequência de execução das várias atividades.

Para definir um problema de natureza combinatória chamamos de X às possíveis soluções e f à função que mede à qualidade das soluções em X. O objectivo é encontrar a solução s* que minimize f entre as X soluções possíveis, que são apenas limitadas pelas restrições do processo. Um problema combinatório pode ser representado por:

$$f(s^*) = \min_{s \in X} f(s)$$

Mesmo com os desenvolvimentos nas ciências computacionais, os problemas de otimização combinatória têm sempre um tamanho máximo para X, para o qual é possível encontrar a solução ótima, em tempo útil. Isso levou ao estudo e desenvolvimento de métodos heurísticos capazes de resolver os problemas combinatórios [7].

3.12. Abordagens aos problemas de escalonamento

Na realidade, a maioria dos problemas de escalonamento estão acompanhados por restrições que tornam a sua resolução intratável à medida que a dimensão do problema aumenta. Muitos problemas de escalonamento, são problemas de optimização combinatória, ou seja, problemas para os quais não existem algoritmos exatos que sejam capazes de calcular a solução óptima, em tempo útil, à medida que variáveis adicionais são acrescentadas ao problema. Para contornar esta complexidade, foram desenvolvidos algoritmos heurísticos capazes de encontrar soluções aceitáveis em tempo útil.

Mesmo com o desenvolvimento de métodos heurísticos, novos métodos exatos, para encontrar as soluções ótimas, continuaram a ser estudados. Estes, continuam a ser usados para resolver problemas de escalonamento polinomiais e a ser utilizados como medidas de comparação ("*benchmarking*") para problemas de natureza exponencial.

Assim, é possível dividir os métodos de resolução entre três tipos de métodos:

Métodos Enumerativos: Onde se encontram as soluções ótimas, através da enumeração e comparação de todas as possíveis soluções. Para problemas mais complexos são métodos demorados, que não permitem encontrar a melhor solução em tempo útil. É importante notar que enumeração é considerada implícita, pois existem maneiras de comparar grupos de soluções sem haver necessidade de as enumerar explicitamente. Entre os métodos enumerativos é possível identificar:

- **Programação Dinâmica** - Criada em 1950 por Bellman. Método que resolve um problema de decisão através de um processo recursivo com múltiplas fases. O problema é dividido em múltiplas fases, onde em cada fase é tomada uma decisão que influencia as decisões nas fases posteriores. A pesquisa pela solução ótima é feita através de uma equação recursiva que compara o ótima da atual fase, com os ótimos da fase anterior. O principio em que Bellman se suporta é: Numa determinada fase o critério de ótimalidade para as fazes posteriores, é independente das decisões tomadas nas fases que a antecedem.

  Embora os algoritmos de programação dinâmica sejam habitualmente de complexidade exponencial, é possível, para muitos problemas *NP-hard* construir algoritmos pseudo-polinomiais, capazes de resolver problemas, com um número razoável de variáveis, sem demasiado esforço computacional.

- ***Branch and Bound*** – É um método de enumeração implícita, isto é, que recorrer a enumeração parcial das soluções. Grupos de soluções cada vez mais pequenos são examinados de maneira encontrar a solução para o problema. Tal como o nome indica o método utilizada dois procedimentos:

  - *Branching*, é o procedimento que divide o problema nos vários sub-problemas que vão ser analisados. É normalmente representado por uma árvore, onde o problema é dividido em vários sub-problemas através dos seus ramos.

  - *Bounding*, é o procedimento que determina o valor limite para a pesquisa na árvore. Esse valor limite, que pode ser encontrado através de métodos heurísticos, serve como critério de eliminação, para os ramos da árvore que serão analisados. Um ramo que tenha uma valor pior do que valor limite, é imediatamente descartado, sem necessitar de ser analisado novamente.

Embora os algoritmos de *Branch and Bound* sejam de natureza exponencial, na pesquisa da solução ótima, podem ser utilizado para encontrar solução eficientes se o procedimento for interrompido depois de determinado tempo.

Métodos Heurísticos: Devido a natureza combinatória de muitos dos problemas de escalonamento foi necessário desenvolver métodos capazes de encontrar soluções satisfatória, em tempo razoável. Assim, os métodos heurísticos permitem desenvolver

algoritmos caracterizados por funções de natureza polinomial. Entre os métodos heurísticos é possível identificar:

- **Heurísticas Geradoras** - Métodos capazes de encontrar uma solução para o problema, através de procedimentos simples. São habitualmente baseados em algoritmos "gulosos", que optam pelo ótimo local em cada nível sem consideração pelos níveis posteriores, sendo por isso também chamados algoritmos "míopes". As soluções encontradas pelas heurísticas geradoras, não são sempre boas, no entanto a simplicidade do método justifica a sua utilização quando há necessidade de tomar a decisão num curto espaço de tempo.

  Este tipo de heurísticas podem ser usadas para encontrar soluções iniciais a que posteriormente se aplicam heurísticas de pesquisa local, ou para auxiliar em métodos de *Branch and Bound*, para encontrar o primeiro limite de eliminação.

- **Heurísticas de Pesquisa Local** – Métodos heurísticos capazes de pesquisar o espaço de soluções de um problema, ou seja, métodos de otimização globais. São chamadas de meta-heurísticas, pois necessitam de heurísticas de decisão mais simples para determinar como a pesquisa é feita. As metas-heurísticas de pesquisa local, consistem num conjunto de estratégias de pesquisa de soluções admissíveis, que pertençam à vizinhança de uma dada solução. Inspiram-se em conceitos de diversas áreas, como: heurísticas clássicas, inteligência artificial, evolução genética, sistemas neuronais, propriedades químicas dos materiais, estatística ou comportamentos do mundo animal.

  Entre as meta-heurísticas é possível encontrar: a Pesquisa Tabu (*Tabu Search*), Arrefecimento Simulado (*Simulated Annealing*) os Algoritmos Genéticos (*Genetic Algorithms*) entre muitas outras.

Regras de Prioridade: Para problemas onde apenas é necessário sequenciar as tarefas é possível recorrer a métodos que atribuem determinadas prioridades às tarefas. Este tipo de regras classifica as tarefas num sequência que irá determinar por que ordem deverão ser executadas. Essa classificação pode ser estática, se a posição das tarefas dentro da sequência não for alterada com o tempo, ou dinâmica, onde as tarefas são sequenciadas sempre que uma decisão deve ser tomada.

Depois da breve explicação dos principais métodos para abordar o problema de escalonamento é possível analisar alguns dos problemas clássicos [7] [9] [10] [12].

3.13. Problemas de escalonamento

Como foi possível verificar anteriormente, a classificação dos problemas de escalonamento através da disposição de recursos na oficina, é muito útil. Nesta secção, vão ser descritos métodos para resolver vários problemas de escalonamento clássicos. Esses métodos vão ser divididos em problemas, em:

- Máquina Única;
- Máquinas Paralelas;
- Linhas de Fabrico;
- Oficinas de fabrico.

Essa divisão vai permitir verificar a utilidade dessa classificação através das semelhanças nas abordagens em problemas que ocorram no mesmo tipo de tipologia. É de notar que os problemas em oficina aberta não serão analisados [7].

3.13.1. Máquina única

Os problema de escalonamento em ambientes de máquina única são os mais simples e permitem muitas vezes encontrar soluções eficientes através de regras de prioridade. O facto de ser possível encontrar soluções eficientes através de regras de aplicação simples, permite utilizar os problemas em ambientes de máquina única como simplificações de problemas de escalonamento em tipologias mais complexas. É possível criar regras de prioridades para minimizar, o tempo medio de execução, o tempo de execução total ponderado, o atraso máximo e o atraso positivo máximo.

Entre os problemas de máquina única serão analisados os problemas de minimização do:

- **Tempo Médio de Execução**: Os problemas de Tempo Médio de Execução ($\bar{F}$) podem ser resolvidos através SPT (*Shortest Processing Time*), que atribui prioridade às tarefas que tenham um tempo de processamento mais curto. Assim, devem ser ordenadas através do p_j crescente, dando prioridade às tarefas que necessitem de menos tempo máquina [6].

- **Tempo de Execução Total Ponderado*:*** Para problemas do tipo $\sum w_j C_j$, que representa o tempo de processamento total, tendo em conta o factor de importância de cada uma das tarefas, pode utiliza-se a regra de prioridades WSPT (*Weighted Shortest Processing Time*). A WSPT ordena as tarefas através da razão w_j/p_j. Se o problema de escalonamento tiver restrições de precedências ($1|prec|\sum w_j C_j$), a eficiência da regra WSPT mantem-se. Para isso as tarefas são dispostas em cadeias, de acordo com as suas precedências e as cadeias são executas por ordem decrescente da razão $\sum w_j/\sum p_j$.

  Existem adaptações da regra WSPT, como o WDSPT (*Weighted Discounted Shortest Processing Time*), que encontra soluções eficientes para o problema ($\sum wj(1-e^{rCj})$), [6].

- **Atraso Máximo**: Para minimizar a atraso máximo (L_{max}), é possível utilizar a regra EDD (*Earliest Due Date*), que ordena as tarefas por ordem crescente das datas de entrega. Uma solução eficiente para a atraso máximo é também uma solução eficiente para o atraso positivo máximo. Assim se for possível utilizar o EDD, para encontrar a solução eficiente para um problema de atraso máximo, a mesma regra produz soluções eficientes para problemas de atraso positivo máximo.

  Um caso particular dos problemas de atraso máximo, são o caso onde nem todas as tarefas estão disponíveis para serem realizadas ao mesmo tempo. Se forem permitidas interrupções ($1|r_j,prmp|L_{max}$), o problema pode ser resolvido através da regra EDD; se não forem, então esse problema terá que ser resolvido pela enumeração parcial das soluções possíveis, através do *Branch-and-Bound*. Mesmo para problemas do tipo $1|r_j|L_{max}$, onde é necessário recorrer à enumeração através do *Branch-and-Bound*, o EDD tem um papel importante, pois é utilizado para determinar que classes de soluções serão descartadas.

  Problemas de atraso máximo, com datas de lançamento e sem interrupções ($1|r_j|L_{max}$), têm sido alvo de uma amplo estudo, pois muitos problemas em Oficina de Fabrico ou Linha de Fabrico, podem ser reduzidos a este tipo de problema [6].

- **Atraso Positivo Total**: Existem casos onde não é possível resolver os problemas através de simples regras de prioridade. Para estes casos, é necessário recorrer à programação dinâmica ou a métodos heurísticos, que resultem em soluções eficientes. Para resolver problema de escalonamento, como o atraso positivo total ($1||\Sigma T_j$), é necessário recorrer a técnicas de resolução mais complexas.

O atraso positivo total, ao contrário do problema de minimização do número de tarefas em atraso, tem em consideração não só as tarefas que não vão cumprir as datas de conclusão, mas também o período do incumprimento. Assim, um problema de atraso positivo total, vai diminuir a possibilidade de uma tarefa ficar em espera muito tempo, mesmo quando essa tarefa não vai cumprir a sua data de conclusão. Para resolver um problema, é necessário recorrer a programação dinâmica, onde se devem enumerar as várias soluções possíveis e comparar os resultados. Existem vários métodos para simplificar o problema, sendo possível dizer que uma tarefa com menor tempo de processamento e uma data de conclusão mais próxima, deve sempre ser processada primeiro [6] [7].

3.13.2. Máquinas paralelas

Nos problemas de máquinas paralelas torna-se adequado dividir o escalonamento da produção entre a afectação e a fase de sequenciação. Essa divisão é clara nos problemas de máquinas paralelas, pois a afectação às máquinas não está dependente da sequência escolhida para as tarefas nas máquinas, como acontece nos problemas em linhas de fabrico ou oficinas de fabrico. Assim, as tarefas deverão ser primeiro afectadas às máquinas, para só depois ser determinada a sequência em que deverão ser executadas.

Entre os problemas habituais em ambientes de máquinas paralelas, vão ser apresentados métodos para abordar os problemas de minimização do:

- *Makespan*;
- Tempo de Execução Total*;*
- Atraso Máximo.

Tal como nos ambientes de máquina única, também nos ambientes de máquinas paralelas podem ser usadas regras de prioridade para encontrar soluções, tanto para problemas de tempo de execução total, como para problemas de minimização do *makespan*. Para os problemas de *makespan*, vai ser necessário introduzir o conceito de caminho crítico, que é uma ferramenta habitual na análise de diagramas de Gantt.

- ***Makespan***: Pode ser dividido em dois tipos: aqueles que permitem interrupções, e aqueles que não permitem. É um problema importante pois permite balancear carga, em cada uma das máquinas.

Os problemas que procuram minimizar o *makespan* sem interrupções e sem relações de precedência ($Pm||C_{max}$), podem ser resolvidos através da heurística LPT (*Longest Processing Time*), onde as tarefas com tempo de processamento mais longo, são executadas primeiro numa das máquinas. Assim que essas tarefas acabem de ser processadas, novas tarefas vão sendo afectadas às máquinas que vão sendo libertadas.

Este tipo de heurística, que atribui prioridade às tarefas através do seu tempo de processamento, vai balancear as cargas nas máquinas com as tarefas de execução mais curta, que aparecem no fim da sequência. O escalonamento através do LPT, produz solução razoáveis, próximas das soluções eficientes, na relação:

$$\frac{\mathrm{Cmax(LPT)}}{\mathrm{Cmax(OPT)}} \leq \frac{4}{3} - \frac{1}{3\mathrm{m}} \qquad (15)$$

Para o caso de existirem relações de precedência no problema ($Pm|prec|C_{max}$), é necessário introduzir o conceito "cadeias de tarefas" e de "caminho crítico". Existem tarefas que podem ser adiadas sem aumentar o *makespan*, como pode ser visto na figura 7. Apenas as tarefas que pertencem ao caminho crítico é que tem de ser processadas assim que disponíveis, para não aumentar o *makespan*.

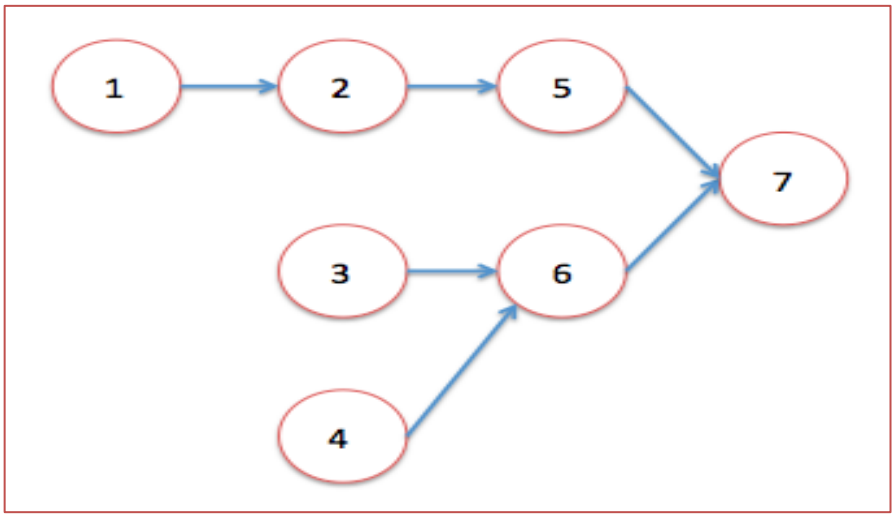

Figura 7 – Caminho Critico

O caminho crítico, que na figura 9 é composto pelas tarefas {1;2;5;7}, representa a cadeia de tarefas com o tempo de processamento mais longo. As tarefas {3;4;6} não fazem parte do caminho crítico e por isso não necessitam de começar a ser executadas assim que estão disponíveis, para minimizar o *makespan*.

Estes problemas são computacionalmente difíceis, com um tempo de processamento que não é limitado por uma função polinomial. No entanto problemas do tipo $Pm|p_j=1,tree|C_{max}$, podem ser resolvido facilmente utilizando a regra CP (*Critical Path*), que dá prioridade às tarefas que pertençam ao caminho crítico, [6]. O CP é equivalente a dar prioridade às tarefas que pertençam a um nível mais elevado. As tarefas de nível 1 são aquelas que não tem sucessores, as de nível 2 são aquelas que têm as tarefas de nível 1 como sucessoras e por ai adiante. Um exemplo dos níveis pode ser analisado na figura 8.

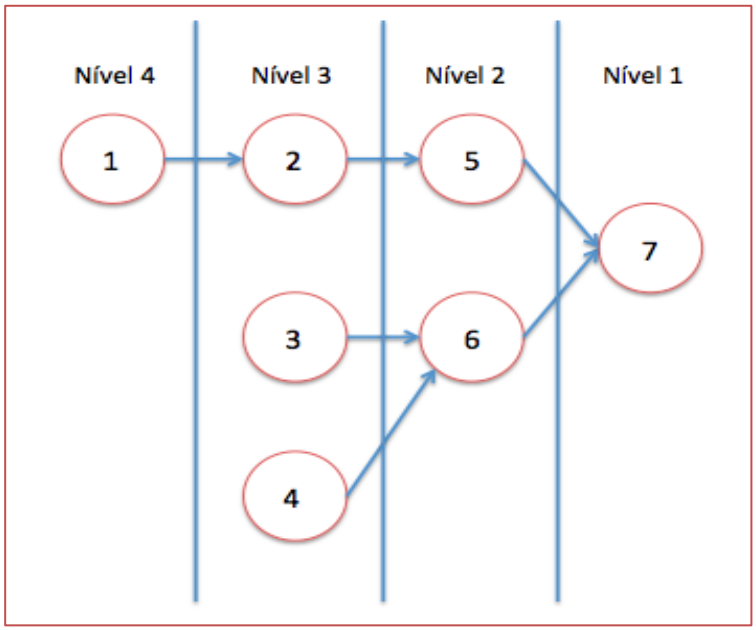

Figura 8 – Hierarquia de Níveis

A CP encontra soluções ótimas para problemas do tipo $Pm|p_j=1,tree|C_{max}$ e pode também ser usada em problemas do tipo $P_2|p_j=1|C_{max}$, com a relação entre a solução ótima e a solução produzida pelo CP:

$$\frac{\text{Cmax(CP)}}{\text{Cmax(OPT)}} \leq \frac{4}{3} \qquad (16)$$

É também possível resolver problemas de minimização do *makespan*, com relações de precedência pela regra LNS (*Largest Number of Sucessors*), que dá prioridade às tarefas que tenham o maior número de sucessores. São considerados não apenas os sucessores diretos, mas também os sucessores dos sucessores. A LNS produz soluções eficientes, tal como o CP, para problemas do tipo $Pm|p_j=1,tree|C_{max}$ [6].

Para o caso de problemas que permitam interrupções ($Pm|prmp|C_{max}$), não é possível utilizar a regra LPT, apresentada anteriormente. Assim, para resolver este tipo de problemas é necessário recorrer a um algoritmo, que encontra as soluções eficientes em tempo, descrito por uma função polinomial. O problema é modelado por:

$$Min\ C_{max}$$

$$s.a.$$

$$\sum_{i=1}^{m} x_{ij} = p_j\,, \qquad j = 1, \dots, n$$

$$\sum_{i=1}^{m} x_{ij} \leq C_{max} \qquad j = 1, \dots, n$$

$$\sum_{j=1}^{m} x_{ij} \leq C_{max} \qquad i = 1, \dots, m$$

$$x_{ij} \geq 0$$

É possível demostrar que o *makespan* terá como limite inferior:

$$C_{max} \geq \left(p_{max}, \sum_{j=1}^{n} \frac{p_j}{m} \right) = C^*_{max} \qquad (17)$$

Tendo em consideração que é possível calcular o limite inferior do *makespan*, o algoritmo na tabela 6 encontra soluções ótimas para $Pm|prmp|C_{max}$:

Tabela 6 – Algoritmo para Minimização do *Makespan* em Máquinas Paralelas

1º Passo – Alocar todas as tarefas numa sequência aleatória a uma das máquinas.
2º Passo – Dividir a sequência em intervalos do tipo $[0, C^*_{max}]$, $[C^*_{max}, 2C^*_{max}]$, $[2C^*_{max}, 3C^*_{max}]$ onde o número de intervalos será igual ao número de máquinas.
3º Passo – Alocar cada um dos intervalos a uma das máquinas.

Por último, é possível utilizar o LRPT (*Longest Remaining Processing Time*), onde as tarefas com mais tempo de processamento necessário a cada momento, têm prioridade. Esta regra gera soluções ótimas, mas quando o período de interrupções é continuo, as soluções recorrem a interrupções infinitas, com as tarefas em execução a serem trocadas indefinidamente [6].

- **Tempo de Execução Total**: Os problemas de minimização do tempo de execução total, em ambiente de máquinas paralelas, pode ser dividido naqueles que permitem interrupções e naqueles que não o permitem. Dentro daqueles que permitem interrupções existem três tipos de problemas: os que procuram minimizar o tempo de execução total sem relações de precedência, os que assumem com relações de precedência entre as tarefas e ainda os que procuram minimizar o tempo de execução total em oficinas onde as máquinas não tem as mesmas características.

Os problemas da classe $Pm||\Sigma C_j$ podem ser resolvidos através da regra SPT, onde a tarefa com o tempo de processamento mais curto vai ser alocada à primeira máquina. A segunda tarefa, com o tempo de processamento mais curto à segundo máquina, até todas as máquinas terem tarefas alocadas. Se o problema tiver pesos associados às tarefas, deverá ser usada a WSPT, embora este não garanta que se encontrem soluções eficientes. No caso de existirem relações de precedência ($Pm|prec,\ tree|\Sigma C_j$), é possível utilizar a regra CP, que dá prioridade às tarefas que pertençam ao caminho crítico, [6]. A regra CP é apenas ótima para problemas de *outtree*, que pode ser analisado na figura 9.

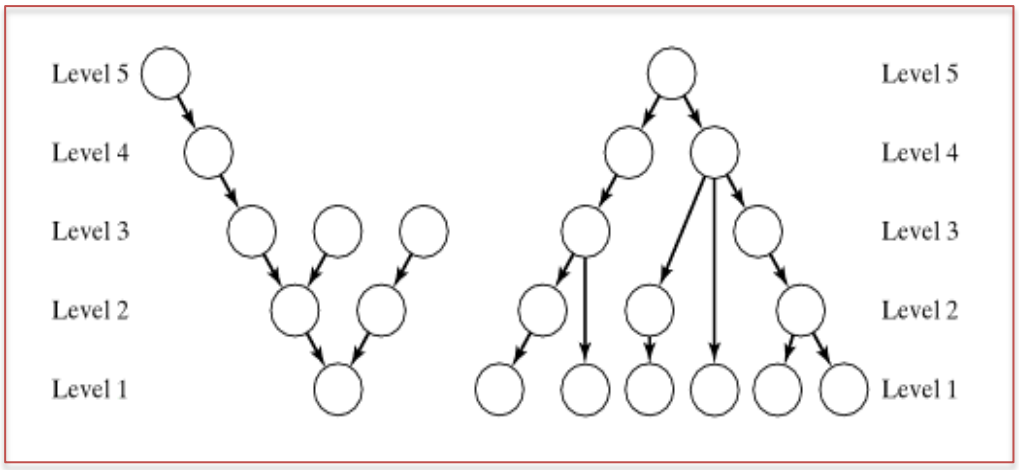

Figura 9 – Intree e Outtree [6]

Para problemas onde as tarefas têm restrições nas máquinas onde podem ser executas (M_j), o problema pode ser resolvido pela regra LFJ (*Least Flexible Job*), onde sempre que uma máquina fica disponível é dada prioridade às tarefas que podem ser processadas pelo menor número de máquinas. No caso das máquinas não serem idênticas é necessário recorrer a programação linear, que deverá encontrar a solução ótima. No caso de serem permitidas interrupções, o problema pode ser resolvido através da regra SRPT-FM (*Shortest Remaining Processing Time on the Fastest Machine*), que não é mais que uma variação do SPT. Assim, em cada momento, a tarefa com o tempo de processamento restante mais longo, deverá ser executada na máquina mais rápida. Como são permitidas interrupções, assim que essa tarefa deixe de necessitar do maior tempo de processamento, a tarefa em execução na segunda máquina mais rápida, deverá ser transferida para a máquina mais rápida.

- **Atraso Máximo**: Problemas de minimização da atraso máximo em ambientes de máquinas paralelas, são tipicamente complexos e não podem ser resolvidos em tempo razoável. No entanto problemas da classe $Qm|prmp|L_{max}$ podem ser resolvidos em tempo polinomial. Para tal é necessário conhecer, à priori, o valor de L_{max}, que vai permitir encontrar a solução, transformando o problema em $Qm|r_j, prmp|L_{max.}$ Essa transformação é realizada atribuindo datas de lançamento às tarefas. Depois da transformação, é possível resolver o problema através da LRPT que foi apresentada anteriormente [6] [7].

3.13.3. Linha de fabrico

Ao contrário dos problemas em máquina única ou máquinas paralelas, os problemas em linha de fabrico (*Flow Shop*), são abordados de maneira diferente, para o caso de haver armazenamento intermédio (ilimitado ou limitado). Problemas com armazenamento inexistente são problemas sequenciais, uma vez que as atividades não podem ser colocadas em espera, pois não existe armazenamento para as atividades em curso de fabrico. Os problemas em linhas de fabrico proporcionais, também apresentam características próprias. Os problemas em linha de fabrico podem ser divididos em:

- Linhas de Fabrico com Armazenamento Ilimitado;
- Linha de Fabrico com Armazenamento Limitado;
- Linhas de Fabrico Proporcionais.

Para os problemas em linha de fabrico com armazenamento ilimitado/limitado serão apresentados métodos para minimizar o *makepsan* e um método para minimizar o tempo de execução total ponderado para os problemas em linhas de fabrico proporcionais [6]:

- ***Makespan* com Armazenamento Ilimitado**: Problemas do tipo $Fm||C_{max}$, não são semelhantes aos problemas do tipo $Fm|prmu|C_{max}$, o que quer dizer que a solução para uma linha de fabrico sequencial, poderá não ser uma solução eficientes para uma linha de fabrico, onde a sequência de tarefas não é a mesma em todas as máquinas. No entanto, para problemas de minimização do *Makespan*, a solução ótima não contempla a troca na sequência das tarefas entre as duas primeiras máquinas e entre as duas máquinas finais. Isso permite concluir que em linhas de fabrico de três máquinas:

$$F3||C_{max} = F3|prmu|C_{max}$$

Para calcular o *Makespan* de uma dada sequência, é possível construir uma rede com as máquinas na horizontal e as tarefas na vertical. Na rede, cada nó deverá conter o tempo de processamento necessário da tarefa k na máquina i e deverá ter arcos de ligação para os nós $(i+1,j_k)$ e (i, j_{k+1}). Depois de construída a rede, é possível transforma-la num diagrama de Gantt, onde o caminho critico é o *Makespan* da sequência. É possível ver um exemplo da transformação de uma rede, num diagrama de Gant na figura 10.

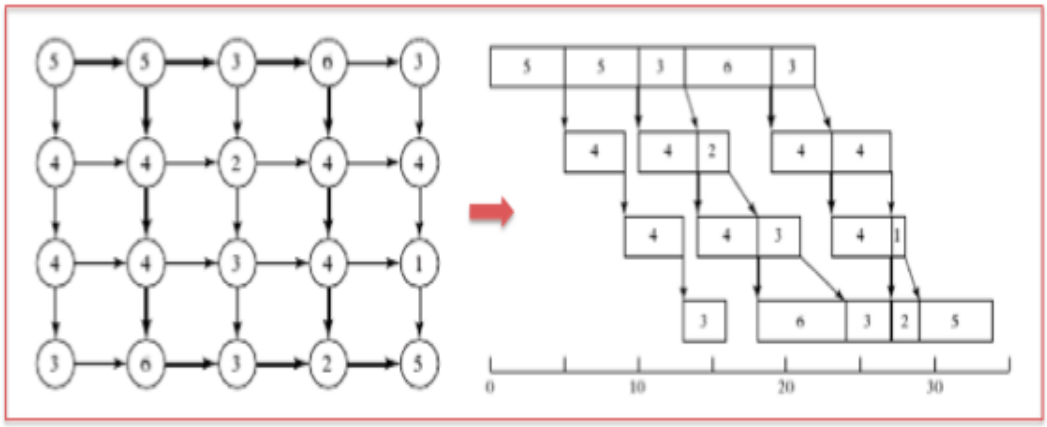

Figura 10 – Rede / Diagrama de Gantt [6]

Para resolver problemas do tipo $F2||Cmax$, é possível utilizar a regra de Johnson, que encontra soluções eficientes, embora possam existir soluções alternativas não complementadas por esta regra.

O algoritmo de Johnson pode ser analisado na tabela 7.

Tabela 7 – Algoritmo de Johnson

1º Passo –As tarefas em que o tempo de processamento na primeira máquina é superior ao tempo de processamento na segunda ($p_{1j}>p_{2j}$), são acrescentadas ao Set I. Aqueles onde o tempo de processamento na segunda máquina e superior ao tempo de processamento na primeira máquina ($p_{1j}<p_{2j}$), são acrescentadas ao Set II. Aquelas tarefas que têm um tempo de processamento semelhante em ambas as máquinas ($p_{1j}=p_{2j}$), podem ser acrescentadas a qualquer um dos Sets.

2º Passo – Ordenar as tarefas que pertencem ao Set I por STP, isto é, por ordem crescente do tempo de processamento.

3º Passo – Ordenar as tarefas que pertencem ao Set II por LTS, isto é, por ordem decrescente do tempo de processamento.

Problemas do tipo $Fm|prmu|C_{max}$, podem ser resolvidos por programação inteira, que procura minimizar o tempo de processamento e o tempo em espera das máquinas. Com x_{jk} a ser uma variável binária que assume o valor de 1 quando a tarefa j está alocada à posição k; I_{ik} representa o tempo de espera na máquina i, entre tarefa k e a tarefa $(k+1)$; W_{ik} o tempo de espera da tarefa k, entre o fim do processamento na máquina i e o começo do processamento na máquina $(i+1)$. É possível modelar o problema por:

$$\text{Min } \sum_{j=1}^{m-1}\sum_{j=1}^{n} x_{j1}\, p_{ij} + \sum_{j=1}^{n-1} I_{mj}$$

s.a.

$$\sum_{j=1}^{n} x_{jk} = 1 \qquad k = 1, \dots, n$$

$$\sum_{k=1}^{n} x_{jk} = 1 \qquad j = 1, \dots, n$$

$$I_{ik} + \sum_{j=1}^{n} x_{j,k+1}\, p_{ij} + W_{i,k+1} - W_{ik} - \sum_{j=1}^{n} x_{jk}\, p_{i+1,j} - I_{i+1,k} = 0$$

$$k = 1, \dots., n-1; \qquad i = 1, \dots, m-1$$

$$W_{i1} = 0 \qquad i = 1, \dots, m\text{-}1$$

$$I_{1k} = 0 \qquad k = 1, \dots, n\text{-}1$$

O problema enunciado não é resolvido em tempo polinomial, sendo por isso de difícil computação. Devido à complexidade do problema, vários métodos heurísticos foram desenvolvidos ao longo dos anos, entre eles a heurística *Slope* [6], que aplica os princípios usados pela regra de Johnson. Esta heurística executa primeiro as tarefas onde o tempo de processamento nas primeiras máquinas é maior do que o tempo de processamento nas últimas. O índice do *Slope* deve ser calculado para cada tarefa, por:

$$A_j = -\sum_{i=1}^{m}\big(m-(2i-1)\big)p_{ij} \qquad (18)$$

Posteriormente as tarefas devem ser ordenadas por ordem decrescente do seu índice de Slope, [6].

- ***Makespan* com Armazenamento Limitado**: Em linhas de fabrico com armazenamento limitado ou não existente, as tarefas não podem ser transferidas para a máquina posterior, se o *buffer* de armazenamento dessa máquina se encontrar cheio. Uma linha de fabrico com armazenamento limitado pode ser simplificada para uma linha de fabrico sem armazenamento, acrescentando máquinas onde as tarefas são executadas instantaneamente. Tal como nos problemas em linhas de fabrico com armazenamento ilimitado, é possível calcular o *Makespan* de uma sequência de tarefas, recorrendo a uma rede que depois é transposta para um diagrama de Gantt. Sendo que neste caso, as tarefas não podem ser empurradas através do sistema, pois podem existir bloqueios que impedem as tarefas de transitar. Esta situação pode ser analisada na figura 11, onde a quarta tarefa não pode começar a ser executada assim que a terceira tarefa termina, devido ao bloqueio que existe na segunda máquina.

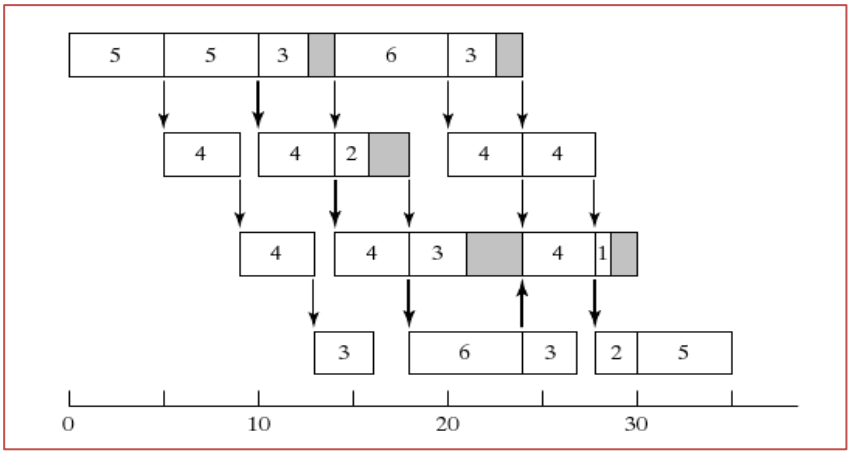

Figura 11 – Diagrama de Gantt para Linha sem Armazenamento [6]

Os problemas da classe $F2|block|C_{max}$, são idênticos ao TSP (*Travelling Salesman Problem*) e ao problema $1|s_{jk}|C_{max}$. Para o problema $1|s_{jk}|C_{max}$ ser considerado idêntico ao TSP o tempo de preparação nas máquina é considerado em duas variáveis diferentes: a primeira representa o estado necessário para começar a executar a tarefa, e a segunda o estado no fim da sua execução. Para o problema $F2|block|C_{max}$ as duas variáveis representam o tempo de processamento na primeira máquina (b_j), e o tempo de processamento na segunda máquina (a_j).

A sequência ótima de tarefas é aquela que minimize a diferença entre a_j é b_{j-1} para evitar bloqueios da linha. Devido às semelhanças entre $1|s_{jk}|C_{max}$ e o $F2|block|C_{max}$, pode ser usado o mesmo método para a resolução de ambos os problemas [6].

Os problemas do tipo $Fm|block|C_{max}$, com mais de duas máquinas não podem ser descritos como problemas TSP e são computacionalmente complexos. Este tipo de problemas pode ser resolvido por métodos heurísticos, como PF (*Profile Fitting*), onde as tarefas são alocadas de maneira a evitar os bloqueios. Sendo $D_{i,j}$ o tempo em que tarefa *j* deixa a máquina *i*. O procedimento da heurística pode ser analisado na tabela 8.

Tabela 8 – Algoritmo da Heurística PF

1º Passo – Escolher a tarefa com o menor tempo de processamento total para iniciar a sequência.
2º Passo – Calcular para todas as tarefas disponíveis os tempos de bloqueio e de espera das máquinas, de acordo com: $$D_{i,j2} - D_{i.j1} - p_{i,j2}$$
3º Passo- Alocar na próxima posição da sequência a tarefa que tenha menor tempos de bloqueios e de espera.
4º Passo- Voltar ao passo dois até todas as tarefas estarem alocada.

Os resultados experimentais da aplicação da heurística PF, demonstram que esta produz resultados próximos do ótimo [6].

- **Tempo de Execução Total Ponderado em Linha de Fabrico Proporcional**: Uma linha de fabrico pode ser designada de proporcional se cada tarefa necessitar do mesmo tempo de processamento em cada uma das máquinas. Os problemas com este tipo de características são bastante semelhantes aos problemas em Máquina Única [6].

O problema $Fm|prmu, p_{ij}=p_j|\sum w_jC_j$ embora continue a ser semelhante ao problema em Máquina Única, não pode ser resolvido da mesma forma. Para encontrar a solução ótima do problema é possível recorrer ao WSPT-MCI (*Weight Shortest Processing Time First with Minimum Cost Insertion*) [6].

Para aplicar o WSPT-MCI, primeiro têm que ser identificadas as tarefas que serão designadas como *new-max*, que são as tarefas que têm um tempo de processamento superior ao de todas as tarefas que a antecedem. A primeira tarefa vai ser sempre considerada *new-max*, pois não tem nenhuma tarefa que a anteceda. Depois de identificadas as tarefas *new-max*, é possível dividir as tarefas em segmentos. Cada um dos segmentos deve começar com uma tarefa *new-max* e conter todas as tarefas até a próxima tarefa *new-max*. Para encontrar a solução ótima, é necessário reordenar as tarefas dentro de cada segmento de acordo com WSPT-MCI.

O WSPT-MCI, também é capaz de encontrar soluções eficientes para problemas do tipo $Fm|p_{ij}=p_j|\sum w_jC_j$, e pode ser adaptado para resolver problemas onde as máquinas não tenham todas a mesma velocidade de processamento [6].

3.13.4. Oficina de fabrico

Entre os problemas abordados, os problemas em oficinas de fabrico (*Job Shop*) são aqueles que têm maior complexidade. Essa complexidade é devida às diferentes características neste tipo de tipologias, onde as tarefas podem percorrer rotas diferentes através do sistema. Os problemas em oficina de fabrico podem ser divididos entre os que:

- Não permitem recirculação;
- Permitem recirculação;

Um problema que permita recirculação, permite uma mesma tarefa visitar a mesma máquina mais do que uma vez. Assim, num problema onde exista recirculação, uma tarefa pode necessitar de uma operação de corte em determinada máquina, continuar o seu processo de fabrico com outras operações, para finalmente regressar à máquina de corte para mais uma operação. Um problema que permita recirculação é tipicamente mais complexo que outro onde cada tarefa apenas pode visitar cada máquina uma única vez [6].

Para problemas em oficinas de fabrico vão ser apresentados dois métodos que minimizam o *makespan*:

- ***Makespan* através de Programação Disjuntiva** : Os problemas do tipo $Jm||C_{max}$, podem ser representados por gráficos disjuntivos. Estes gráficos são constituídos por um número de nós (*N*) igual ao total de operações do problema. Os nós, devem estar conectados por arcos conjuntivos (*A*) e disjuntivos (B). Os arcos conjuntivos representam as rotas das tarefas através do sistema. Os arcos disjuntivos são representados por linhas a tracejado e ligam duas tarefas que tem que ser processadas na mesma máquina. Ambos os arcos devem ter representado o tempo de processamento da operação que representam. Par além dos N nós que representam todas as operações, deve ainda existir um nó de fonte (*U*) e um nó de destino (*V*). Os gráfico disjuntivos são representados por *G*=(*N*,*A*,*B*).

  Um exemplo de um gráfico disjuntivo pode ser visto na figura 12, para um problema de três tarefas e quatro máquinas.

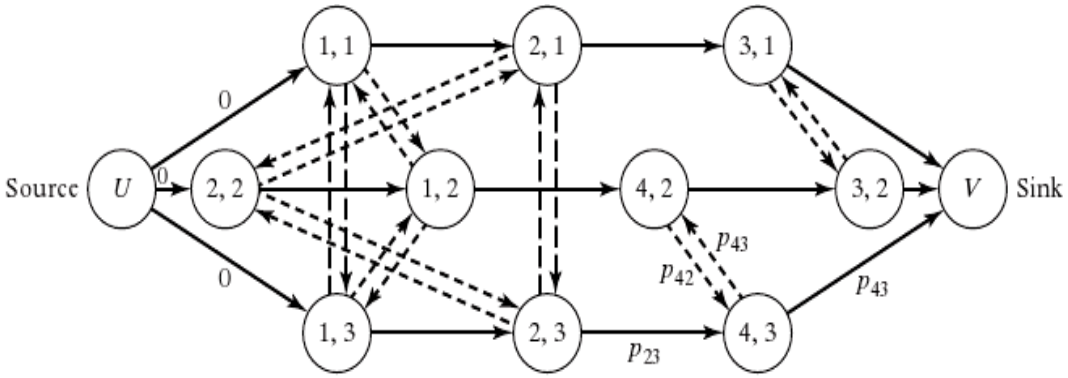

Figura 12 – Gráfico Disjuntivo [6]

Para encontrar a solução do problema é necessário determinar qual a sequência das tarefas nas máquinas, o que corresponde a determinar o arco disjuntivo que selecionar em cada par. A solução ótima é aquela que minimiza o caminho crítico entre a fonte e o destino. O problema pode ser formulado em programação disjuntiva. Este tipo de programação divide as restrições em conjuntivas e disjuntivas. As restrições conjuntivas devem ser sempre satisfeitas. Enquanto no grupo de restrições disjuntivas apenas uma das restrições tem que ser satisfeita [6].

Com y_{ij}, a representar quando a tarefa *j*, começa a ser executada na máquina *i* e p_{ij}, o tempo de processamento da tarefa *j*, na máquina *i*, é possível formular o problema de minimização do *Makespan* da seguinte maneira:

Min C_{max}

s.a.

$y_{kj} - y_{ij} \geq p_{ij}$	para todos $(i,j) \rightarrow (k,j) \in A$
$C_{max} - y_{ij} \geq p_{ij}$	para todos $(i,j) \in N$
$y_{ij} - y_{il} \geq p_{il}$ ou $y_{il} - y_{ij} \geq p_{ij}$	para todos (i,l) e (i,j), $i = 1, ..., m$
$y_{ij} \geq p_{il}$	para todos $(i,j) \in N$

Embora o problema possa ser formulado assim, para encontrar as soluções ótimas é necessário recorrer a métodos exatos, como por exemplo de *Branch-and-Bound*, que apenas vai ter em consideração as soluções ativas. É possível encontrar todas as soluções ativas em três passos [6], como é possível analisar na tabela 9.

Tabela 9 – Algoritmo para Gerar as Soluções Ativas

1º Passo - Ω contem as primeiras operações de todas as tarefas;

$r_{ij} = 0$, para todos $(i,j) \in \Omega$.

2º Passo – Calcular para escalonamento parcial:

$t(\Omega) = \min_{(i,J) \in \Omega} \{r_{ij} + p_{ij}\}$;

A máquina que apresente o mínimo é representada por i*;

3º Passo - Ω' contem todas as operações (i^*, j) onde:

$r_{i^*,j} < t(\Omega)$;

Considerar todas as operações em Ω' como a próxima na maquina i* ;

Remover essa operação de Ω e adicionar a operação posterior;

Voltar ao 2º Passo.

O problema vai sequenciar as tarefas nas máquinas que vão executar mais que uma tarefa, como um problema $1|r_j|L_{max}$, que não pode ser resolvido em tempo polinomial. O facto de ser necessário resolver vários problemas computacionalmente difíceis, para encontrar a solução ótima para problemas de minimização do *Makespan* em Oficinas de Fabrico, levou ao desenvolvimento de métodos heurísticos [6] [7].

- ***Shifting Botleneck*:** É uma heurística bastante utilizada para resolver problemas de minimização do *makespan* em ambientes de Oficina de Fabrico. É baseada no principio de *bottleneck* circulante, onde a máquina onde ocorre o *bottleneck*, é escolhida para ser tratada individualmente. Depois de escolhida a sequência em que as tarefas vão ser executadas nessa máquina, é escolhida outra maquina, até todas as máquinas serem analisadas individualmente [6].

  Embora a heurística apresente bons resultados, continua a ser necessário resolver problemas da classe $1|r_j|L_{max}$, para cada uma das máquinas individualmente. O procedimento da heurística do *Shifting Bottleneck* pode ser analisado na tabela 10.

Tabela 10 – Algoritmo da Heurística *Shifting Bottleneck*

1º Passo - M = Todas as máquinas no problema;
2ª Passo - M_0 = Máquinas onde as tarefas já foram sequenciadas;
Inicialmente $M_0 = \varnothing$;
3º Passo – Escolher entre M – M_0 a máquina *bottleneck*. Determinar qual vai ser o próximo *bottleneck*, resolver um problema tipo $1\|r_j\|L_{max}$, para cada uma das máquinas e escolher aquela que obteve o atraso máximo;
4º Passo - Para a máquina escolhida como *bottleneck*, conectar os arcos disjuntivos, de maneira a minimizar a atraso máximo;
5º Passo - Para todas as máquinas pertencentes a M_0, analisar a sequência das tarefas novamente;
6º Passo - Até todas as máquinas pertencerem a M_0, voltar ao 3º passo.

Com pequenas variações a heurística do *Shifting Bottleneck*, pode ser utilizada para resolver problemas de atraso positivo total pesado ($Jm||\Sigma w_j T_j$). Nesse caso o *bottleneck* é a máquina onde aconteça o pior atraso positivo total pesado ($1||\Sigma w_j T_j$), que pode ser tratado pela regra de prioridade ATC (*Apparent Tardiness Cost*) [6].

Uma variante desta heurística pode ainda ser usada para problemas de minimização do *makespan*, em linhas de fabrico flexíveis com recirculação [6] [7].

3.14. Conclusão

Neste o capítulo foi abordado o problema de escalonamento da produção. O problema de escalonamento pode ser divido em duas fases distintas, a afectação das operações aos recursos disponíveis e a sequenciação das tarefas, de forma a melhor utilizar os recursos, cumprir prazos e otimizar os tempos de produção. O problema de escalonamento está maioritariamente relacionado com problemas de produção, sendo por isso utilizada uma denominação semelhante à utilizada numa organização fabril.

Num problema clássico de escalonamento os diferentes elementos são denominados por:

- **Máquinas** - Os recursos disponíveis;
- **Tarefas** - Uma trabalho localizado no tempo;
- **Operações** - Uma parte de tarefa que apenas utiliza um recurso;
- **Oficina** - Local onde estão dispostos os recursos afectos a produção.

O objectivo do escalonamento da produção está relacionado com a produtividade da oficina, a pontualidade das tarefas e a rentabilização dos recursos e é necessário definir métodos capazes de avaliar o desempenho das soluções encontradas. No entanto, o estudo académico do escalonamento ainda apresentam diferenças significativas dos problemas encontrados na realidade industrial.

Entre os métodos usados foi possível analisar métodos exatos que são pouco apropriados para problemas com funções de complexidade não polinomiais e os métodos heurísticos que embora não garantam as soluções ótimas, requerem menos tempo de processamento. As heurísticas são particularmente importantes para problemas de otimização combinatória, isto é, problemas de escalonamento onde a complexidade aumenta exponencialmente à medida que a dimensão dos problemas aumenta.

Foi apresentada a denominação que vai ser usada no trabalho para descrever os problemas de escalonamento para, por fim, apresentar vários modelos e métodos para a resolução de problemas de escalonamento, em oficinas a operar com apenas uma máquina, com máquinas paralelas, em linhas de fabrico e em oficinas de fabrico.

4. PROBLEMAS DE ESCALONAMENTO COM MÁQUINAS PARALELAS

Durante este capítulo vai ser abordado, em maior detalhe, o problema da minimização do *makespan*, em máquinas paralelas. No início, vão ser descritas as principais restrições, nos problemas de máquinas paralelas, para posteriormente rever os métodos existentes para a sua resolução. Finalmente, vai ser realizada uma análise das principais características dos métodos descritos.

4.1. Introdução

O primeiro problema de minimização do *makespan* em máquinas paralelas, é o problema sem qualquer tipo de restrição ($P_m||C_{max}$), descrito no capítulo anterior. O problema é *NP-Hard*, sendo equivalente ao problema *PARTITION*, que é uma simplificação do problema da mochila (*Knapsack*), onde o beneficio de um item (w_j), é igual ao volume que ocupa na mochila (p_j). Assim é necessário recorrer a métodos heurísticos, para encontrar uma solução eficiente nos problemas mais complexos. Existem vários métodos heurísticos descritos na literatura para resolver este problema [6].

O problema de minimização do *makespan* em máquinas paralelas, sem qualquer tipo restrição, pode ser modelado da seguinte forma:

$$Min\ C_{max}$$

s.a:

$$\sum_{i=1}^{m} x_{ij} = 1\,, \qquad j = 1, \dots, n$$

$$\sum_{j=1}^{n} p_j\, x_{ij} \leq C_{max} \qquad i = 1, \dots, m$$

$$x_{ij} \geq 0$$

$$X_{ij} = \begin{cases} 1\ se\ a\ tarefa\ estiver\ alocada\ \text{à}\ máquina\ j \\ 0\ se\ a\ tarefa\ não\ estiver\ alocada\ \text{à}\ máquina\ j \end{cases}$$

Este tipo de problema pode ser resolvido pela regra de prioridade LTP (*Longest Processing Time*), onde as tarefas com maior tempo de processamento têm prioridade. Assim, ao afectar primeiro as tarefas com maior tempo de processamento, é possível minimizar os desvios com as últimas tarefas, que são as mais curtas. Aplicado a regra LTP, é possível obter um resultado próximo do ótimo, com a relação de:

$$\frac{C_{max}(LTP)}{C_{max}(OPT)} \leq \frac{4}{3} - \frac{1}{3m} \qquad (19)$$

O próximo problema é aquele que procura minimizar o *makespan* para problemas de máquinas paralelas quando existirem interrupções ($P_m|prmp|C_{max}$), descrito no último capítulo. Habitualmente, o recurso a interrupções, simplifica o problema, que assim pode ser resolvido otimamente por um algoritmo polinomial. É possível encontrar a solução ótimas, pois é possível calcular o *makespan* mínimo para só posteriormente alocar as tarefas às máquinas. É também possível utilizar a regra LRPT (*Longest Remaining Processing Time*), para a resolução deste problema.

O problema de minimização do *makespan* em máquinas paralelas com interrupções, pode ser modelado da seguinte forma:

$$Min\ C_{max}$$

s.a:

$$\sum_{i=1}^{m} x_{ij} = p_j\,, \qquad j = 1, \dots, n$$

$$\sum_{i=1}^{m} x_{ij} \leq C_{max} \qquad j = 1, \dots, n$$

$$\sum_{j=1}^{n} x_{ij} \leq C_{max} \qquad i = 1, \dots, m$$

$$x_{ij} \geq 0$$

Assim, como é possível dividir a execução de uma tarefa por várias máquinas, o *makespan* terá como limite inferior:

$$C_{max} \geq \left(p_{max}, \sum_{j=1}^{n} \frac{p_j}{m} \right) = C^*_{max} \qquad (20)$$

Conhecido o limite inferior para o *makespan* máximo, torne-se possível utilizar o algoritmo na tabela 11, para encontrar a solução ótimas:

Tabela 11 – Algoritmo para Minimização do *Makespan* com Interrupções

1º Passo – Alocar todas as tarefas numa sequência aleatória a qualquer máquinas.
2º Passo – Dividir a sequência em intervalos do tipo $[0, C^*_{max}]$, $[C^*_{max}, 2C^*_{max}]$, $[2C^*_{max}, 3C^*_{max}]$ onde o número de intervalos será igual ao número de máquinas.
3º Passo – Alocar cada um dos intervalos a uma máquina.

O problema em máquinas diferentes ($R_m||C_{max}$), representa um caso particular dos problemas com máquinas paralelas. Neste caso, as máquinas disponíveis apresentam características distintas, sendo os tempos de processamento das tarefas, diferentes para cada uma. Isto é o que acontece quando, por exemplo, existem vários computadores com configurações diferentes e várias tarefas, com requisitos diferentes, para serem executadas (CPU, GPU, Memória).

O problema de minimização do *makespan* em ambientes com máquinas diferentes, dispostas em paralelo, pode ser modelado da seguinte forma:

$$Min\ C_{max}$$

s.a:

$$\sum_{i=1}^{m} x_{ij} = 1\,, \qquad j = 1, \dots, n$$

$$\sum_{j=1}^{n} p_{ij}\, x_{ij} \leq C_{max} \qquad i = 1, \dots, m$$

$$x_{ij} \geq 0$$

$$X_{ij} = \begin{cases} 1\ se\ a\ tarefa\ estiver\ alocada\ \text{à}\ máquina\ j \\ 0\ se\ a\ tarefa\ não\ estiver\ alocada\ \text{à}\ máquina\ j \end{cases}$$

Para o problema de máquinas diferentes em paralelo ($R_m||C_{max}$), existem várias heurísticas que permitem encontrar soluções próximas da ótima [6] [13] [14].

4.2. Métodos heurísticos

Neste secção serão apresentadas as heurísticas (MET, MCT, Min-Min, K-Percent Best, SWA, Surffage), para resolução do problema em máquinas diferentes, ($R_m||C_{max}$). A descrição será acompanhada por um exemplo ilustrativo [15] [16] [17].

O problema teste será semelhante para todas as heurísticas, permitindo identificar as vantagens e limitações de cada uma e pode ser visto na tabela 12.

Tabela 12 – Problema Teste

	Máquina 1	Máquina 2	Máquina 3
Tarefa 1	50	20	50
Tarefa 2	70	80	70
Tarefa 3	50	70	50
Tarefa 4	10	30	20

4.2.1. MET (*Minimum Execution Time*)

A heurística MET, utiliza o tempo de processamento para afectar as tarefas. Isso leva a afectação de cada tarefa isoladamente, isto é, sem tomar em consideração as outras tarefas na escolha da máquina onde será executada. A heurística MET pode ser analisada na tabela 13.

Tabela 13 – Algoritmo da Heurística MET

1º Passo – As tarefas são colocadas, numa lista em ordem aleatória.
2º Passo – A primeira tarefa é alocada à máquina com o menor tempo de processamento.
3º Passo – A tarefa alocada em 2º passo é removida da lista de tarefas.
4º Passo – Os passo 2 e 3 são repetidos até todas as tarefas terem sido alocadas.

O procedimento afecta as tarefas numa ordem aleatória, o que não é importante pois a afectação de uma tarefa não tem em consideração as outras tarefas já afectadas. Pode parecer que o facto de apenas ser considerado o tempo de processamento, não produz os melhores resultados, particularmente em problemas com muitas tarefas, no entanto a simplicidade da heurística permite afectar as tarefas muito rapidamente.

Na tabela 14, é possível analisar a simulação do procedimento da heurística MET na resolução do problema teste, com a afectação assinalada a amarelo.

Tabela 14 – Simulação da Heurística MET

	Máquina 1	Máquina 2	Máquina 3
Tarefa 1	50	20	50
Tarefa 2	70	80	70
Tarefa 3	50	70	50
Tarefa 4	10	30	20

No exemplo, a tarefa 1 é alocada à máquina 2, onde apresentou um tempo de processamento de 20. A tarefa 2, pode ser alocada tanto à máquina 1 como à máquina 3, onde tem um tempo de processamento de 70. A tarefa 3, também pode ser alocada a duas máquinas, à máquina 1 ou à máquina 3, com um tempo de execução de 50. A última tarefa é alocada na máquina 1, onde apresentou um tempo de processamento de 10. O plano de escalonamento das tarefas pode ser analisado na figura 13.

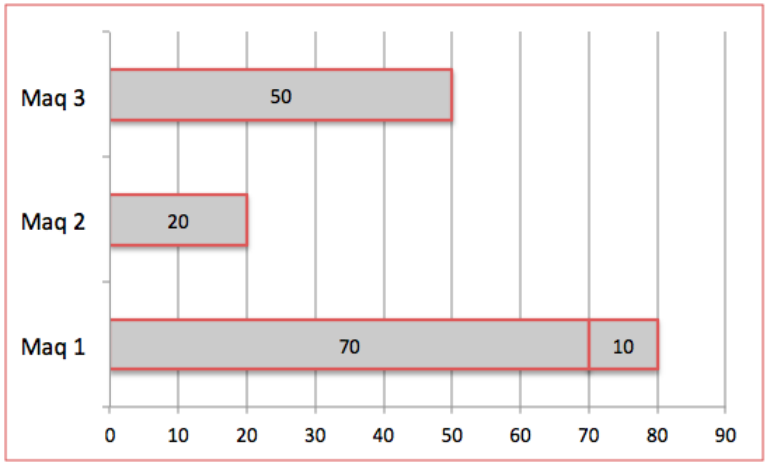

Figura 13 – Solução da Heurística MET

As duas tarefas (tarefa 2 e tarefa 3) que podem ser afectadas a duas máquinas diferentes foram resolvidas aleatoriamente, resultando num *makespan* de 80 *u.t.*; é de notar que como os empates foram resolvidos aleatoriamente, era possível obter um *makespan* ótimo de 70 *u.t.* ou um *makespan* ou 130 *u.t.* [15] [16] [17].

4.2.2. MCT (*Minimum Completion Time*)

A heurística MCT utiliza o tempo de conclusão para afectar as tarefas, ao contrário da heurística anterior que usa o tempo de processamento, tendo em consideração as tarefas já alocadas às máquinas. Para isso é necessário utilizar uma matriz com as datas de conclusão, que deve ser atualizada sempre que uma tarefa é afectada a uma das máquinas.

O procedimento da heurística MCT pode ser analisada na tabela 15:

Tabela 15 – Algoritmo da Heurística MCT

1º Passo – As tarefas são colocadas numa lista em ordem aleatória.
2º Passo – A primeira tarefa é alocada à máquina com a menor data de conclusão.
3º Passo – A tarefa alocada em 2º passo é removida da lista das tarefas.
4º Passo – A matriz com as datas de conclusão das tarefas é atualizada.
5º Passo – Os passo 2, 3 e 4 são repetidos até todas as tarefas terem sido alocadas.

Na tabela 16, é possível analisar a simulação do procedimento da heurística MCT, assumindo que a ordem das tarefas não foi alterada, na resolução do problema teste.

Tabela 16 – Simulação da Heurística MCT

	Máquina 1	Máquina 2	Máquina 3
Tarefa 1	50	20	50
Tarefa 2	70	80	70
Tarefa 3	50	70	50
Tarefa 4	10	30	20
Tarefa 2	70	100	70
Tarefa 3	50	90	50
Tarefa 4	10	50	20
Tarefa 3	120	90	50
Tarefa4	80	50	20
Tarefa 4	80	50	70

Como é possível constatar, para a primeira tarefa a data de conclusão é igual ao seu tempo de processamento. Isso pode não acontecer quando uma ou mais máquinas não estão disponíveis desde do inicio. Neste caso todas as máquinas estão disponíveis desde o momento zero, sendo por isso a tarefa 1, afecta à máquina onde têm o menor tempo de processamento, que é a máquina 2, com um tempo de processamento de 20. Para afectar a próxima tarefa vai ser necessário atualizar a disponibilidade das máquinas, sendo para isso necessário somar 20, a todas as datas de conclusão na máquina 2, que vai ter que executar a tarefa 1, para além de qualquer outra tarefa.

Existe um empate na afectação da tarefa 2, que pode ser afecta na máquina 1 ou na máquina 3, tendo em ambas uma data de conclusão de 70. As disponibilidades têm que novamente ser atualizadas antes de se afectar a tarefa 3, que é afecta à máquina 3. A última tarefa é afecta à máquina 2. O *makespan* total é de 70 *u.t.* e embora tenham existido empates, estes não tem qualquer influência no *makespan* total. No caso da ordem de afetação das tarefas ter sido outra, o *makespan* podia ser 80 *u.t.*. O plano de escalonamento das tarefas pode ser analisado na figura 14 [15] [16] [17].

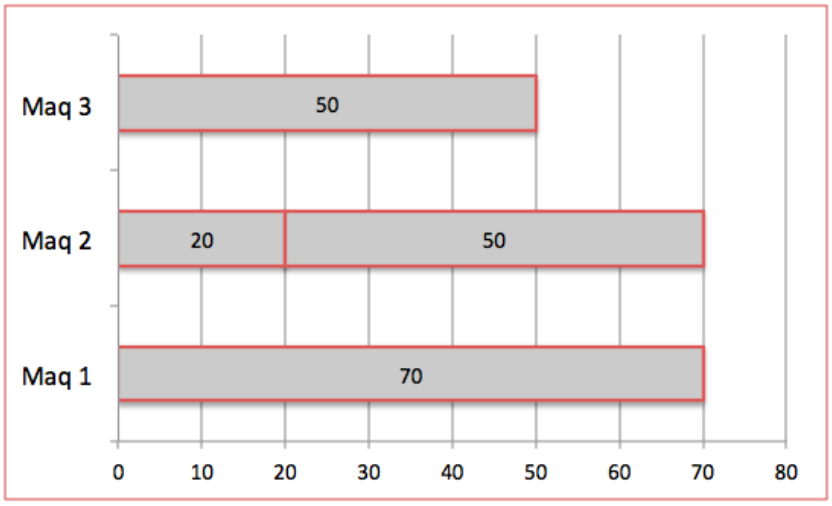

Figura 14 – Solução da Heurística MCT

4.2.3. Min-Min

A heurística Min-Min, utiliza a datas de conclusão mínima para determinar a que máquina a tarefa vai ser afecta e posteriormente escolhe a tarefa com menor data de conclusão para ser afecta primeiro.

O procedimento da heurística pode ser analisado na tabela 17.

Tabela 17 – Algoritmo da Heurística Min-Min

1º Passo – As tarefas são colocadas numa lista em ordem aleatória.

2º Passo – Para todas tarefas é determinada a máquina com menor data de conclusão.

3º Passo – Entre todos os pares tarefa/máquina encontrados no 2º passo, é determinado aquele que têm a menor data de conclusão.

4º Passo – O par tarefa/máquina determinado no 3º passo é afecto e a tarefa é removida da liste da tarefas.

5º Passo – A matriz com os tempos de conclusão das tarefas é atualizada.

6º Passo – Os passo 2, 3, 4 e 5 são repetidos até todas as tarefas terem sido afectas.

Na tabela 18, é possível analisar a simulação do procedimento da heurística Min-Min, na resolução do problema teste.

Tabela 18 – Simulação da Heurística Min-Min

	Maq 1	Maq 2	Maq 3
Tarefa 1	50	20	50
Tarefa 2	70	80	70
Tarefa 3	50	70	50
Tarefa 4	10	30	20
Tarefa 1	60	20	50
Tarefa 2	80	80	70
Tarefa 3	60	70	50
Tarefa 2	80	100	70
Tarefa 3	60	90	50
Tarefa 2	80	100	120

Na matriz os pares tarefa/máquinas estão assinalados a vermelho com a escolha final assinalada a amarelo. A tarefa 4, é afectada à máquina 1 e uma vez atualizadas as disponibilidades, é afectada a tarefa 1, à máquina 2, com uma data de conclusão de 20. Depois da matriz ser novamente atualizada a tarefa 3, é afectada a máquina 3, com um data de conclusão de 50. A ultima tarefa é afectada à máquina 1. O plano de escalonamento é semelhante ao da heurística MET e pode ser analisado na figura 13.

O *makespan* total é de 80 *u.t.*, pior que o resultado encontrado pela heurística MTC e semelhante a um dos resultados possível da heurística MET. Embora tenham existido empates nas tarefas 2 e 3, na afectação da primeira tarefa, estes não tiverem qualquer influência no resultado. Com a heurística Mim-Min, a ordem das tarefas não interfere no resultado final do *makespan* [15] [16] [17].

4.2.4. K-Percent Best

A heurística K-Percent Best, utiliza o tempo de processamento, e a data de conclusão, para afectar as tarefas. Esta heurística é um misto da heurística MET e da heurística MCT, tentando complementar a simplicidade da primeira com o melhor desempenho da segunda. O procedimento da heurística pode ser analisado na tabela 19.

Tabela 19 – Algoritmo da Heurística K-Percent Best

1º Passo – As tarefas são colocadas numa lista em ordem aleatória.
2º Passo – Para a primeira tarefa da lista são selecionadas as $m\frac{k}{100}$ máquinas que tenham o menor tempo de processamento.
3º Passo – Entre as máquinas selecionadas no 2º passo, a tarefa é alocada aquela onde têm a menor data de conclusão.
4º Passo – A tarefa é removida da lista e os datas de conclusão são atualizadas.
5º Passo – Os passos 2, 3 e 4 são repetidos até todas as tarefas terem sido alocadas.

O valor de *k* deve ser determinado por experimentação. É de notar que a definição do valor *k*, vai determinar o quão próxima a heurística vai estar de MET ou de MCT:

$$k = \begin{cases} 100 \ heurística \text{ é } semelhante \ a \ \ MCT \\ 100/_{m} \ heurística \text{ é } semelhante \ a \ MET \end{cases}$$

Assim, no exemplo apresentado, será considerado um *k*=70, o que vai permitir selecionar as duas máquinas que tenham o menor tempo de processamento, como é possível verificar:

$$3\frac{70}{100} \approx 2$$

O problema exemplo, vai ser resolvido através da heurística K-Percent Best, assumindo que a ordem das tarefas não é alterada. A alocação pode ser visto na tabela 20.

Tabela 20 – Simulação da Heurística K-Percent Best

	Máquina 1	Máquina 2	Máquina 3
Tarefa 1	50	20	-
Tarefa 2	70	-	70
Tarefa 3	50	-	50
Tarefa 4	10	-	20
Tarefa 2	70	-	70
Tarefa 3	50	-	50
Tarefa 4	10	-	20
Tarefa 3	120	-	50
Tarefa 4	80	-	20
Tarefa 4	80	-	70

Para cada tarefa apenas são consideradas as duas máquinas com o menor tempo de processamento. Dentro das máquinas selecionáveis, as tarefas vão ser afectas à máquina com a menor data de conclusão. Assim, a tarefa 1 é afecta à máquina 2 e as disponibilidades atualizadas. Na tarefa 2 existe novamente um empate, que não irá afectar o *makespan* total e será novamente resolvido de forma aleatória, afectado a tarefa 2 à máquina 1. As disponibilidades são novamente atualizadas, somando 70 *u.t.*, da afectação da tarefa 2, à máquina 1. Posteriormente é afecta a tarefa 3 à máquina 3. A matriz tem que ser novamente atualizada, antes de afecta a ultima tarefa à máquina 3.

O plano de escalonamento das tarefas pode ser analisado na figura 15.

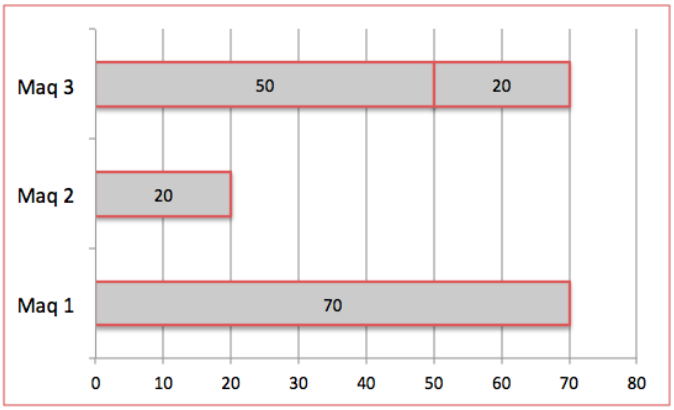

Figura 15 – Solução da Heurística K-Percent Best

O *makespan* total é de 70 *u.t.*, que é semelhante ao encontrado pela heurística MTC. Embora tenham existido empates na tarefas 2, estes não tiverem qualquer influência no *makespan* total do K-Percent Best [15] [16] [17].

4.2.5. SWA (*Switching Algorithm*)

A heurística SWA é um misto das heurísticas MET e da heurística MCT, como acontece com a heurística K-Percent Best, sendo que SWA ainda está mais próximo das heurísticas MET e MCT, pois apenas utiliza um algoritmo para calcular um valor de *threshold*, que vai determinar qual das duas heurísticas vai utilizar. A heurística procura um compromisso entre a simplicidade da heurística MET, com o melhor desempenho da heurística MCT. O procedimento da SWA pode ser analisado na tabela 21.

Tabela 21 – Algoritmo da Heurística SWA

1º Passo – As tarefas são colocadas numa lista em ordem aleatória.
2º Passo – A primeira tarefa é aloca a uma máquina através da heurística MCT.
3º Passo – É calculado o índice *Load Balance*.
4º Passo – A heurística utilizada para alocar a próxima tarefa é: i) Se Load Balance ≥ "high threshold" é utilizado MET; ii) Se Load Balance ≤ "low threshold" é utilizado MCT; iii) Se "high threshold" ≥ Load Balance ≥ "low threshold" a heurística da tarefa anterior continua a ser utilizada.
5º Passo – Os passos 3 e 4 são repetidos até todas as tarefas terem sido alocadas.

Os valores "*high threshold*" e "*low threshold*" devem ser definidos experimentalmente. O valor do *Load Balance* é calculado através da expressão:

$$\frac{Ready\ Time\ mínimo\ entre\ todas\ as\ máquinas}{Ready\ Time\ máximo\ entre\ todas\ as\ máquinas} \qquad (21)$$

Assim, no exemplo apresentado será considerado um:

High Threshold = 0.8

Low Threshold = 0.7

Na tabela 22, é possível analisar a simulação do procedimento da heurística SWA, na resolução do problema teste.

Tabela 22 – Simulação da Heurística SWA

	Maq 1	Maq 2	Maq 3	Rea 1	Rea 2	Rea 3	LB	Heurística
Tarefa 1	50	20	50	0	0	0	-	MCT
Tarefa 2	70	80	70	0	0	0	-	-
Tarefa 3	50	70	50	0	0	0	-	-
Tarefa 4	10	30	20	0	0	0	-	-
Tarefa 2	70	100	70	0	20	0	0	MCT
Tarefa 3	50	70	50	0	20	0	-	-
Tarefa 4	10	30	20	0	20	0	-	-
Tarefa 3	120	90	50	70	20	0	0	MCT
Tarefa 3	10	30	20	70	20	0	-	-
Tarefa 2	80	50	70	70	20	50	0.28	MCT

Como todas as tarefas estão disponíveis no momento zero, a primeira tarefa é afecta à máquina onde apresentou o menor tempo de processamento. Para afectar a próxima tarefa é necessário calcular o *Load Balance* e compará-lo com os valores de "*threshold*", para determinar que heurística vai ser usada.

Neste caso o índice é menor que o "*low threshold*" sendo, por isso, a tarefa afecta através da heurística MCT, afectando a tarefa 2, à máquina 1. O índice LB da tarefa 3, é novamente menor que "low threshold", levando à escolha da heurística MCT. Deve ainda ser referido que não era necessário o índice LB ser menor que "*low threshold*" para utilizar a heurística MCT, se o índice estivesse entre os dois valores de "*threshold*" a heurística usada para a última tarefa mantinha-se inalterada. A última tarefa é afecta, utilizando a heurística MCT, à maquina 2, onde tem um data de conclusão de 50. Note-se que a matriz não foi sempre atualizada, apenas os tempos de processamento das tarefa que estavam a ser afectadas foram atualizados, porque as tarefas posteriores podiam ser afectas através da heurística MET.

O plano de escalonamento das tarefas pode ser analisado na figura 16.

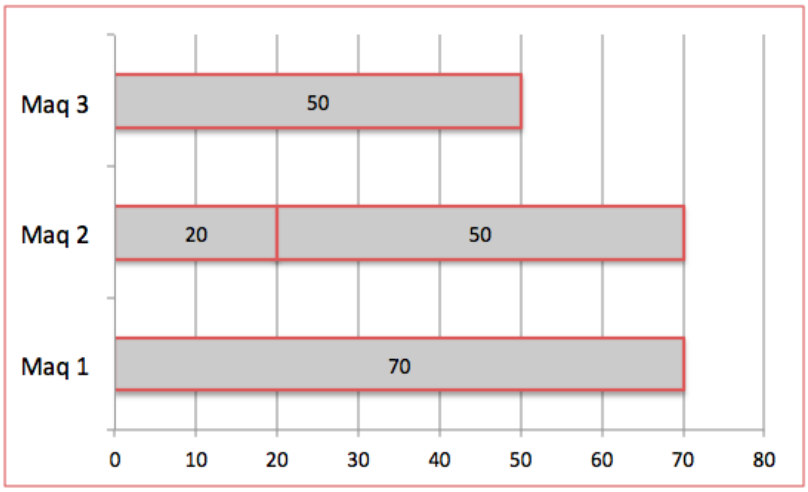

Figura 16 – Solução da Heurística SWA

O *makespan* total é de 70 *u.t.*, tal como aconteceu com heurística MCT, com a mesma sequência de tarefas. Se a sequência fosse alterada, era possível obter um *makespan* total de 80, o que não corresponde à solução ótima do problema [15] [16] [17].

4.2.6. Suffrage

A heurística Suffrage afecta as tarefas às máquinas com a menor data de conclusão, sendo possível alocar mais que uma tarefa de uma vez. No caso de existirem duas ou mais tarefas a competir pela mesma máquina, é dada prioridade à que tenha maior diferença nas datas de conclusão entre as duas melhores máquinas. O propósito é alocar primeiro as tarefas que mais aumentem o *makespan* total, quando não associadas à melhor máquina.

O procedimento da heurística Suffrage, pode ser analisado na tabela 23.

Tabela 23 – Algoritmo da Heurística Suffrage

1º Passo – As tarefas são colocadas numa lista em ordem aleatória.

2º Passo – Enquanto houver tarefas não alocadas:

i) Para cada máquina encontrar as tarefas que tenham a menor data de conclusão nessa máquina:

a) Se apenas existir uma tarefa, alocar à tarefa a essa máquina e remove-la da lista de tarefas por alocar.

b) Se existir mais que uma tarefa, alocar à tarefa com maior índice de S e remove-la da lista de tarefas.

i) Atualizar a matriz de disponibilidade de cada uma das máquinas.

O índice S pode ser calculado:

$$S = Second\ Smallest\ Completion\ Time - Smallest\ Completion\ Time \quad (22)$$

Na tabela 24, é possível analisar a simulação do procedimento da heurística Suffrage, na resolução do problema teste.

Tabela 24 – Simulação da Heurística Suffrage

	Máquina 1	Máquina 2	Máquina 3	S
Tarefa 1	50	20	50	30
Tarefa 2	70	80	70	0
Tarefa 3	50	70	50	0
Tarefa 4	10	30	20	10
Tarefa 3	60	90	120	30

Durante a primeira iteração foram afectas três tarefas. A tarefa 1 foi afectada à máquina 2. Nesta fase, existem três tarefas a concorrer pela máquina 1, sendo-lhe afecta a tarefa 4, que apresenta o maior valor de S. Nesta fase, existem duas tarefas que concorrem pela máquina 3, neste caso o índice S é igual para ambas tendo sido afecta a tarefa 2, de forma aleatória. Finalmente, a última tarefa é alocada à maquina 1.

O plano de escalonamento das tarefas pode ser analisado na figura 17.

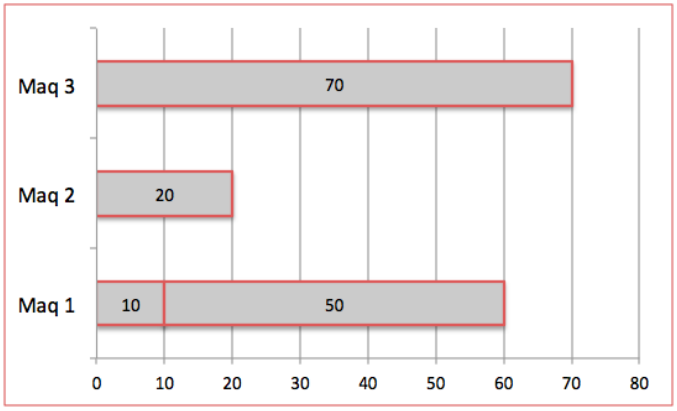

Figura 17 – Solução da Heurística Suffrage

A solução apresenta um *makespan* de 70 *u.t.*. É de notar que se o empate entre tarefas tivesse sido resolvido de outro modo, o *makespan* seria 80 *u.t.*. Nas heurísticas MCT, Min-Min e SWA, foram necessárias quatro matrizes para calcular a solução, com heurística Suffrage, apenas foram necessárias duas para encontrar a solução final [15] [16] [17].

4.3. Conclusão

Nas heurísticas apresentadas é possível identificar características comuns a várias delas. As heurísticas MET e MCT, parecem servir como base para as restantes. A primeira parece a heurística mais simples, resolvendo o problema através de uma única matriz que não precisa de ser atualizada, no entanto ao apenas tomar em consideração os tempos de processamento, pode levar à afectação de todas as tarefas à mesma máquina.

A heurística MCT é um pouco mais complexa, pois utiliza as datas de conclusão para afectar as tarefas. Isso requer a atualização das disponibilidades sempre que uma tarefa é afecta a uma máquina. Esta heurística é muito influenciada pela ordem com que as tarefas são afectas. A heurística Min-Min, utiliza as datas de conclusão. São utilizadas para determinar qual a sequência da afectação, e posteriormente para escolher a que máquina as tarefas serão afectas. As duas heurísticas, K-Percent Best e SWA, são um misto das heurísticas MET e MCT. O propósito, é aliar a simplicidade da heurística MET com os melhor desempenho da heurística MCT.

Com a heurística Suffrage podem ser afectas várias tarefas de cada vez, com cada uma a concorrer pela melhor máquina. O desempate é analisado através da possível perda de tempo das tarefas, ao não ser afectas à máquina preferida.

Na tabela 25 é possível analisar as soluções obtidos no problema teste.

Tabela 25 – Soluções das Heurísticas

Heurísticas	*Makespan* com sequencia fixa	*Makespan* com qualquer sequencia
MET	70 / 80 / 130	70 / 80 / 130
MCT	70	80 / 90
Min-Min	80	-
K-Percent	70	80
SWA	70	80
Suffrage	70 / 80	-

Como foi possível verificar, nenhuma heurística encontrou sempre a solução ótima do problema. Várias heurísticas encontram a solução ótima no problema teste, no entanto considerando que a sequência de resolução é aleatória, não é possível afirmar que qualquer uma das heurísticas vai produzir sempre um escalonamento ótimo.

5. HEURÍSTICAS PROPOSTAS

Neste capítulo, serão propostas heurísticas de afectação de tarefas a recursos, tendo como critério de otimização a redução do *makespan*.

5.1. Introdução

No capítulo anterior foram analisadas heurísticas descritas na literatura, usadas para resolver problemas de minimização do *makespan* em máquinas diferentes ($R_m||C_{max}$). Foi possível identificar as suas principais limitações, como o facto de a maioria estar muito dependente da sequência pela qual as tarefas são afectas ou o desempate entre máquinas ser resolvido aleatoriamente. O facto da heurística Suffrage alocar mais que uma tarefa numa única iteração pode, também, influenciar o seu desempenho.

No sentido de melhorar o desempenho das heurísticas, são propostas versões modificadas das heurísticas MCT e Suffrage. Pretende-se tirar partido das vantagens das heurísticas e modifica-las para reduzir as limitações identificadas. O seu funcionamento será descrito nas secções seguintes, utilizando problemas ilustrativos.

Serão ainda descrito o método de implementação da ferramentas informática de apoio ao estudo computacional, implementado no âmbito deste trabalho de mestrado. Foram implementadas duas das heurísticas propostas e as duas heurísticas não modificadas. Deve referir-se que as heurísticas de afectação foram codificadas na linguagem de programação C, no Microsoft Visual Studio 2012.

5.2. OMCT 1 (*Ordered Minimum Completion Time 1*)

A heurística OMCT 1, utiliza um cálculo similar ao do índice S da heurística Suffrage para ordenar as tarefas, antes de afectar as tarefas através das datas de conclusão. O procedimento da heurística OMCT 1, pode ser analisado na tabela 26:

Tabela 26 – Algoritmo da Heurística OMCT 1

1º Passo – Calcular o índice S para todas as tarefas.

2º Passo – Ordenar as tarefas por ordem decrescente do índice S.

3º Passo – A primeira tarefa é alocada à máquina com o menor data de conclusão.

4º Passo – A tarefa é removida da lista e as datas de conclusão são atualizadas.

5º Passo – Os passos 3 e 4 são repetidos até todas as tarefas terem sido alocadas.

Serão apresentados dois exemplos onde é possível verificar as diferenças entre a OMCT 1 a heurística MCT e a heurística Suffrage. O primeiro exemplo vai demostrar o procedimento da OMCT 1 e permitir verificar as diferenças entre esta e a heurística MCT e pode ser analisado na tabela 27.

Tabela 27 – Simulação da Heurística OMCT 1

	Máquina 1	Máquina 2	S
Tarefa 1	100	110	10
Tarefa 2	100	120	20
Tarefa 2	100	120	20
Tarefa 1	100	110	10
Tarefa 1	200	110	10

Makespan = 110

O resultado do mesmo problema através da heurística MCT pode ser visto na tabela 28.

Tabela 28 – 2º Simulação da Heurística MCT

	Máquina 1	Máquina 2
Tarefa 1	100	110
Tarefa 2	100	120
Tarefa 1	200	120

Makespan = 120

Como é possível constatar a OMCT 1, encontra a solução ótima do problema, enquanto a heurística MCT não. Tal deve-se ao facto da OMCT 1, ordenar as tarefas antes de aplicar o algoritmo de afectação, enquanto MCT afecta as tarefas numa sequência aleatória. O plano de escalonamento de ambas as heurísticas pode ser analisado na figura 18.

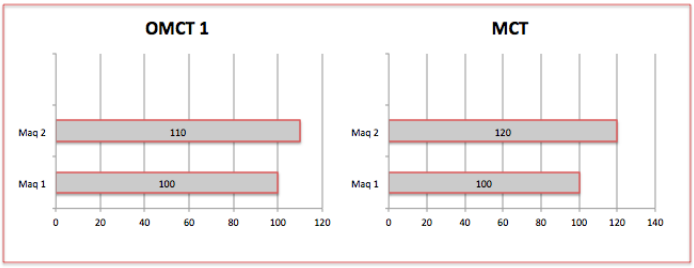

Figura 18 – Solução das Heurísticas OMCT 1 e MCT

O próximo exemplo vai permitir analisar as diferenças entre a heurística OMCT 1 e a heurística Suffrage. O resultado pode ser analisado na tabela 29.

Tabela 29 – 2º Simulação da Heurística OMCT 1

	Máquina 1	Máquina 2	Máquina 3	S
Tarefa 1	100	120	500	20
Tarefa 2	100	140	500	40
Tarefa 3	500	100	110	10
Tarefa 2	100	140	500	40
Tarefa 1	100	120	500	20
Tarefa 3	500	100	110	10
Tarefa 1	200	120	500	20
Tarefa 3	600	100	110	10
Tarefa 3	600	220	110	10

Makespan = 120 u.t.

O resultado da heurística Suffrage pode ser analisado na tabela 30.

Tabela 30 – 2º Simulação da Heurística Suffrage

	Máquina 1	Máquina 2	Máquina 3	S
Tarefa 1	100	120	500	20
Tarefa 2	100	140	500	40
Tarefa 3	500	100	110	10
Tarefa 3	200	220	500	20

Makespan = 200 u.t.

O plano de escalonamento de ambas as heurísticas pode ser analisado na figura 19.

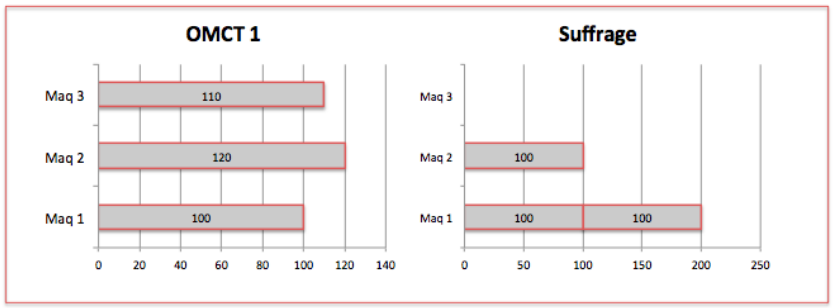

Figura 19 – Solução das Heurísticas OMCT 1 e Suffrage

A heurística OMCT 1, encontra a solução ótima, enquanto a heurística Suffrage não. Tal deve-se ao facto da heurística Suffrage alocar mais do que uma tarefa em cada iteração.

É possível identificar situações onde não é possível encontrar a solução ótima através da heurística OMCT 1. Uma desses situações é apresentado na tabela 31.

Tabela 31 – Limitação da Heurística OMCT 1

	Máquina 1	Máquina 2	Máquina 3	S
Tarefa 1	100	150	260	50
Tarefa 2	100	130	270	30
Tarefa 3	200	300	400	100
Tarefa 4	160	100	300	60
Tarefa 3	200	300	400	100
Tarefa 4	160	100	300	60
Tarefa 1	100	150	260	50
Tarefa 2	100	130	270	30
Tarefa 4	360	100	300	60
Tarefa 1	300	150	260	50
Tarefa 2	300	130	270	30
Tarefa 1	300	250	260	50
Tarefa 2	300	130	270	30
Tarefa 2	300	380	270	30

Makespan = 270 u.t.

O *makespan* ótimo era 260 *u.t.*, com a afectação da tarefa 1 à máquina 3 e da tarefa 2 à máquina 2. O problema nasce do facto de apenas serem consideradas as duas melhores máquinas, para cada tarefa, no cálculo do índice S. Este facto não permite ter em consideração as outras máquinas na atribuição das prioridades e uma das tarefas é afecta à máquina mais lenta. Este problema é particularmente relevante quando a maioria das tarefas têm o menor tempo de processamento na mesma máquina, ou quando existe uma dispersão muito grande entre os tempos de processamento de uma tarefa.

5.2. OMCT 2 (*Ordered Minimum Completion Time 2*)

A heurística OMCT 2, vai utilizar o desvio padrão para ordenar as tarefas. Isso permite sequenciar as tarefas através da dispersão das datas de conclusão em todas as máquinas, ao contrário da heurística anterior, que apenas tem em consideração as duas melhores máquinas. Ao utilizar todas as máquinas no cálculo do índice de prioridade, não se assume que as tarefas vão ser afectas uma das duas máquinas com menor data de conclusão. O procedimento da heurística OMCT 2, pode ser analisado na tabela 32.

Tabela 32 – Algoritmo da Heurística OMCT 2

1º Passo – Calcular o desvio padrão dos tempos de processamento de todas as tarefas.
2º Passo – Ordenar as tarefas por ordem decrescente do desvio padrão.
3º Passo – A primeira tarefa é afecta à máquina com o menor data de conclusão.
4º Passo – A tarefa é removida da lista e as datas de conclusão são atualizadas.
5º Passo – Os passos 3 e 4 são repetidos até todas as tarefas terem sido afectas.

O desvio padrão pode ser calculado pela equação:

$$\sigma = \sqrt{\frac{\sum(X - \bar{X})^2}{n}} \qquad (23)$$

O problema de escalonamento onde a heurística OMCT 1 não encontrou a solução ótima (tabela 31), vai ser utilizado para demostrar o procedimento da heurística OMCT 2. Isso vai permitir analisar as diferenças no desempenho das heurísticas, num problema onde três das quatro tarefas concorrem para ser alocadas à mesma máquina.

O resultado pode analisado na tabela 33.

Tabela 33 – Simulação da Heurística OMCT 2

	Máquina 1	Máquina 2	Máquina 3	σ (≈)
Tarefa 1	100	150	260	67
Tarefa 2	100	130	270	74
Tarefa 3	200	300	400	82
Tarefa 4	160	100	300	84
Tarefa 4	160	100	300	84
Tarefa 3	200	300	400	82
Tarefa 2	100	130	270	74
Tarefa 1	100	150	260	67
Tarefa 3	200	400	400	82
Tarefa 2	100	230	270	74
Tarefa 1	100	250	260	67
Tarefa 2	300	230	270	74
Tarefa 1	300	250	260	67
Tarefa 1	300	380	260	67

Makespan = 260 u.t.

Utilizando a heurística OMCT 2, foi possível encontrar a solução ótima para o problema. O procedimento alterou a ordem de afectação das tarefas, o facto de ter afectado a tarefa 2 antes de afectar a tarefa 1, permitiu obter uma solução com um *makespan* de 260 *u.t.*.

O plano de escalonamento de ambas as heurísticas pode ser analisado na figura 20.

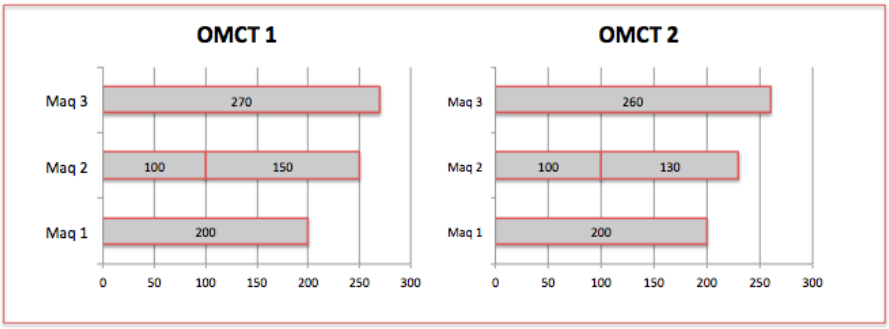

Figura 20 – Solução das Heurísticas OMCT 1 e OMCT 2

É possível identificar situações onde não é possível encontrar a solução ótima através da heurística OMCT 2. Uma desses situações é apresentado na tabela 34.

Tabela 34 – Limitação da Heurística OMCT 2

	Máquina 1	Máquina 2	Máquina 3	σ (≈)
Tarefa 1	100	120	260	71
Tarefa 2	100	170	270	70
Tarefa 3	200	300	400	82
Tarefa 4	160	80	300	91
Tarefa 4	160	80	300	91
Tarefa 3	200	300	400	82
Tarefa 1	100	120	260	71
Tarefa 2	100	170	270	70
Tarefa 3	200	380	400	82
Tarefa 1	100	200	260	71
Tarefa 2	300	250	270	70
Tarefa 1	300	200	260	71
Tarefa 2	500	250	270	70
Tarefa 2	500	370	270	70

Makespan = 270 u.t.

Neste problema a tarefa 1, embora tenha maior dispersão dos tempos de processamento, tem tempos de processamento sempre menores que a tarefa 2. Assim, com a heurística OMCT 2, a tarefa 1 têm prioridade sobre a tarefa 2, embora a afectação da última tenha sempre um maior impacto no *makespan* total do problema.

O mesmo problema (tabela 34) resolvido utilizando heurística OMCT 1, resulta num *makespan* menor, como é possível analisar na tabela 35.

Tabela 35 – 3º Simulação da Heurística OMCT 1

	Máquina 1	Máquina 2	Máquina 3	S
Tarefa 1	100	120	260	20
Tarefa 2	100	170	270	70
Tarefa 3	200	300	400	100
Tarefa 4	160	80	300	80
Tarefa 3	200	300	400	100
Tarefa 4	160	80	300	80
Tarefa 2	100	170	270	70
Tarefa 1	100	120	260	20
Tarefa 4	360	80	300	80
Tarefa 2	300	170	270	70
Tarefa 1	300	120	260	20
Tarefa 2	300	250	270	70
Tarefa 1	300	200	260	20
Tarefa 1	300	370	260	20

Makespan = 260 u.t.

No problema apresentado (tabelas 34 e 35), a heurística OMCT 2, encontra uma solução pior que a heurística OMCT 1, que encontrou a solução ótima. O plano de escalonamento de ambas as heurísticas pode ser analisado na figura 21.

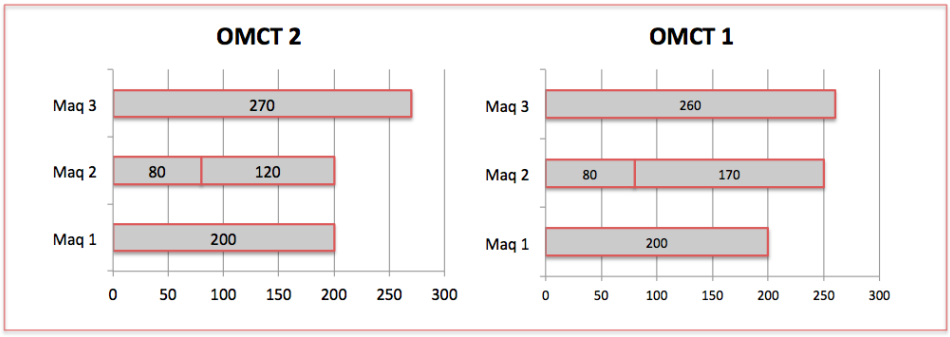

Figura 21 – Solução das Heurísticas OMCT 2 e OMCT 1

5.4. OMCT 3 (*Ordered Minimum Completion Time 3*)

Depois de analisadas ambas as heurísticas é possível identificar as limitações de cada uma. A heurística OMCT 1, parece apresentar melhores soluções em problemas em que não existe muita dispersão nos tempos de processamento e as máquinas mais rápidas para cada tarefa estão melhor distribuídas. A heurística OMCT 2, parece ser melhor adaptada para os problemas onde não exista muita diversidade nos tempos de processamento e há muitas tarefas a concorrem pelas mesmas máquinas.

A heurística OMCT 3, vai utilizar uma soma ponderada entre o índice S e o desvio padrão para decidir porque sequência as tarefas vão ser afectadas. Vai atribuir prioridade às tarefas que tenham uma maior dispersão nos tempos de execução e, ao mesmo tempo, atribuir maior importância às duas máquinas mais rápidas de cada tarefa. Como é uma soma ponderada, é possível adaptar a heurística, experimentalmente, para problemas diferentes. O procedimento da heurística OMCT 3, pode ser analisado na tabela 36.

Tabela 36 – Algoritmo da Heurística OMCT 3

1º Passo – Calcular do índice P dos tempos de execução de todas as tarefas.
2º Passo – Ordenar as tarefas por ordem decrescente do índice P.
3º Passo – A primeira tarefa é alocada à máquina com o menor data de conclusão.
4º Passo – A tarefa é removida da lista e as datas de conclusão são atualizadas.
5º Passo – Os passos 3 e 4 são repetidos até todas as tarefas terem sido alocadas.

Assim as tarefas vão ser afectadas segundo a sequência determinada pela equação:

$$P = a * \sigma + (1 - a) * S \qquad a \ \epsilon [0; 1] \qquad (24)$$

No calculo P existem três casos onde a OMCT 3 tem características particulares. Esses três casos estão dependentes do valores que *a* pode tomar:

- Quando $a = 0$; o método é semelhante a OMCT 1;
- Quando $a = 1$; o método é semelhante a OMCT 2;
- Quando $a = 0.5$; quando é atribuída a mesma importância ao índice S e a dispersão dos tempos de execução;

Na tabela 37, é possível analisar o procedimento da heurística OMCT 3, com $a = 0.25$, no problema em que a heurística OMCT 1, não encontrou a afectação ótima (tabela 31).

Tabela 37 – Simulação da Heurística OMCT 3

	Máquina 1	Máquina 2	Máquina 3	P (≈)
Tarefa 1	100	150	260	62
Tarefa 2	100	130	270	63
Tarefa 3	200	300	400	86
Tarefa 4	160	100	300	78
Tarefa 3	200	300	400	86
Tarefa 4	160	100	300	78
Tarefa 2	100	130	270	63
Tarefa 1	100	150	260	62
Tarefa 4	360	100	300	78
Tarefa 2	300	130	270	63
Tarefa 1	300	150	260	62
Tarefa 2	300	230	270	63
Tarefa 1	300	250	260	62
Tarefa 1	300	380	260	62

Makespan = 260 u.t.

O plano de escalonamento de ambas as heurísticas pode ser analisado na figura 22.

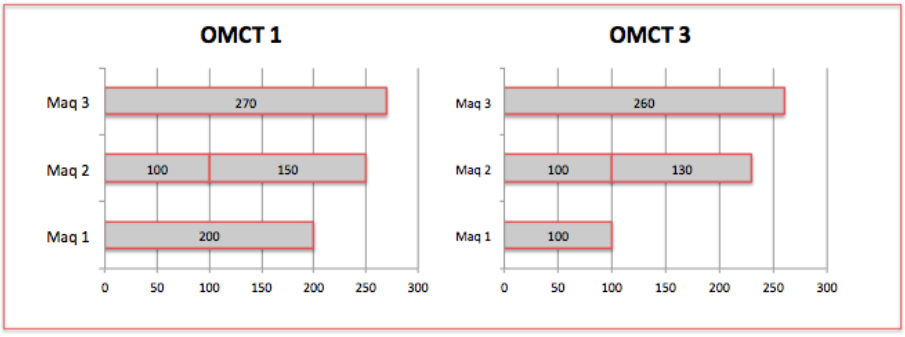

Figura 22 – Solução das Heurísticas OMCT 1 e OMCT 3

Na tabela 38, é possível analisar o procedimento da heurística OMCT 3, com $a = 0.5$, no problema em que a heurística OMCT 2, não encontrou a afectação ótima (tabela 34).

Tabela 38 – 2º Simulação da Heurística OMCT 3

	Maquina 1	Maquina 2	Maquina 3	P (≈)
Tarefa 1	100	120	260	46
Tarefa 2	100	170	270	70
Tarefa 3	200	300	400	91
Tarefa 4	160	80	300	86
Tarefa 3	200	300	400	91
Tarefa 4	160	80	300	86
Tarefa 2	100	170	270	70
Tarefa 1	100	120	260	46
Tarefa 4	360	80	300	86
Tarefa 2	300	170	270	70
Tarefa 1	300	120	260	46
Tarefa 2	300	250	270	70
Tarefa 1	300	200	260	46
Tarefa 1	500	370	260	46

Makespan = 260 u.t.

O plano de escalonamento de ambas as heurísticas pode ser analisado na figura 23.

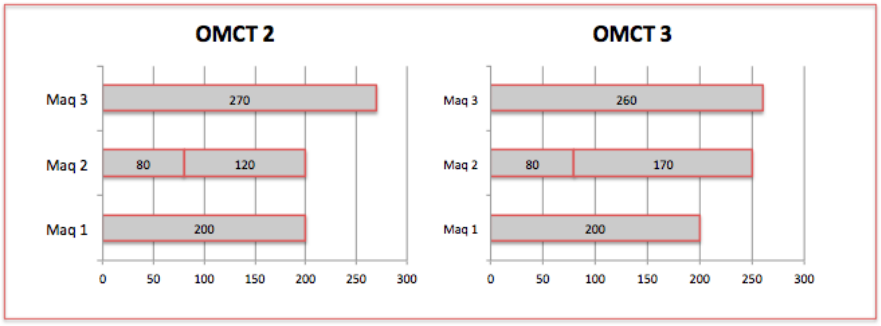

Figura 23 – Solução das Heurísticas OMCT 2 e OMCT 3

Como foi possível analisar a heurística OMCT 3, encontrou sempre a solução ótima, mesmo nos problemas demostrativos das limitações das heurística OMCT 1 e OMCT 2.

5.5. Suffrage One

Nesta secção é proposta uma heurística baseada na Suffrage. Como foi possível verificar, no exemplo apresentado no último capítulo, o facto da heurística Suffrage alocar várias tarefas simultaneamente, levou a um resultado não ótimo. Assim, propõem-se uma variação da heurística Suffrage, de maneira a apenas ser alocada uma tarefa em cada iteração. O procedimento da Suffrage One pode ser analisado na tabela 39.

Tabela 39 – Algoritmo da Suffrage One

1º Passo – Calcular o índice S para todas as tarefas.

2º Passo – Ordenar as tarefas por ordem decrescente do índice S.

3º Passo – A primeira tarefa é afecta à máquina com o menor data de conclusão.

4º Passo – A tarefa é removida da lista e as datas de conclusão são atualizadas.

5º Passo – Os passos 1, 2, 3 e 4 são repetidos até todas as tarefas terem sido afectadas.

O procedimento da Suffrage One pode ser analisado na tabela 40.

Tabela 40 – Simulação da Heurística Suffrage One

	Máquina 1	Máquina 2	Máquina 3	S
Tarefa 1	100	130	400	30
Tarefa 2	100	140	300	40
Tarefa 3	120	100	130	20
Tarefa 2	100	140	300	40
Tarefa 1	100	130	400	30
Tarefa 3	120	100	130	20
Tarefa 1	200	130	400	70
Tarefa 1	220	100	130	30
Tarefa 3	230	230	130	100

Makespan = 130 u.t.

O plano de escalonamento das tarefas pode ser analisado na figura 24.

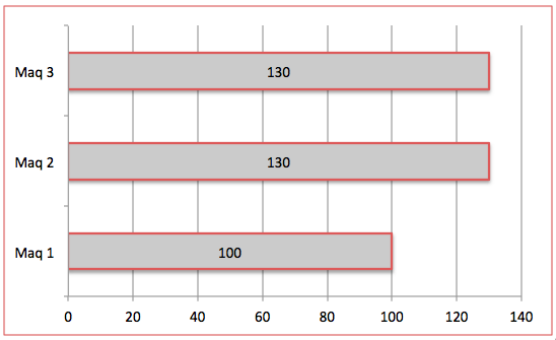

Figura 24 – Solução da Heurística Suffrage One

O mesmo problema resolvido através da heurística Suffrage, sem a alteração proposta resulta num *makespan* de 200 *u.t.*. Isso deve-se ao facto de na primeira iteração serem alocadas duas tarefas, a tarefa 2 à máquina 1 e a tarefa 3 à máquina 2.

Na tabela 41, é possível analisar o procedimento da heurística Suffrage One, no problema em que a heurística OMCT 1, não encontrou a afectação ótima (tabela 31).

Tabela 41 – 2º Simulação da Heurística Suffrage One

	Máquina 1	Máquina 2	Máquina 3	S
Tarefa 1	100	150	260	50
Tarefa 2	100	130	270	30
Tarefa 3	200	300	400	100
Tarefa 4	160	100	300	60
Tarefa 3	200	300	400	100
Tarefa 4	160	100	300	60
Tarefa 1	100	150	260	50
Tarefa 2	100	130	270	30
Tarefa 4	360	100	300	200
Tarefa 2	300	130	270	140
Tarefa 1	300	150	260	110
Tarefa 2	300	230	270	40
Tarefa 1	300	250	260	10
Tarefa 1	300	380	260	40

Makespan = 260

Tanto a heurística OMCT 3, como à heurística Suffrage On encontraram a solução ótima do problema, como pode ser analisado nas tabelas 37 e 41.

5.6. Implementação das heurísticas de afectação

Tendo como objectivo avaliar o desempenho das heurísticas propostas foi desenvolvida uma ferramenta informática. Neste sentido foram implementadas duas heurísticas referidas na literatura, a MCT [15] [16] [17], e a Suffrage [15] [16] [17], que devem servir para fazer uma análise de desempenho das heurísticas propostas. Entre as heurísticas propostas foram implementadas as heurísticas OMCT 3 e a heurística Suffrage One. As heurísticas OMCT 1 e OMCT 2, são consideradas casos particulares da heurística OMCT 3 e por isso não foram implementadas. A heurística MCT, foi implementada para resolver o problema de escalonamento numa ordem pré-definida.

As heurísticas foram implementadas na linguagem de programação C, utilizando o Microsoft Visual Studio 2012. Os resultados são apresentados em duas matrizes; a primeira preenchida com 0 e 1, mostra como as tarefas foram alocadas; a segunda mostra a ocupação de cada uma das máquinas. O *makespan* total corresponde à máquina que tenha maior taxa de ocupação. O código das heurísticas implementadas encontra-se em anexo.

5.6.1. Implementação da heurística MCT

A heurística, MCT, é a mais simples de implementar. A matriz *Ex*, deve ser iniciada com os tempos de execução de cada tarefa. O vector *Ds*, mostra as disponibilidade das máquinas. O pseudo-código pode ser analisado na tabela 42.

Tabela 42 – Pseudo-Código da Heurística MCT

```
INICIO
    PARA X de 0 até J
    ENCONTRAR Min em Ex[I,X] + Ds[I]
    Ds[Min] = Ds[Min] + Ex[Min,X]
        PARA Y de 0 até I
            SE Y=MIN
            Ex[Y,X] = 1
            FIM SE
            SENÃO
            Ex[Y,X] = 0
            FIM SENÃO
        FIM PARA
    FIM PARA
Cmax = Ds[Max]
FIM
```

É utilizado um ciclo que vai percorrer todas as tarefas para determinar qual é a máquina que assegura a menor data de conclusão. Assume-se que a sequência em que as tarefas forem introduzidas é a mesma porque vão ser alocadas.

5.6.2. Implementação da heurística Suffrage

Na heurística Suffrage são, novamente, utilizadas as matrizes *Ex* e *Ds*. Foram ainda utilizados três vectores, S, J e T, que servem para calcular o índice S e determinar que tarefas vão ser afectas em cada iteração. Pseudo-código pode ser analisado na tabela 43.

Tabela 43 – Pseudo-Código da Heurística Suffrage

```
INICIO
    ENQUANTO existirem tarefas por alocar
    S[J] = 0
            PARA X de 0 até J
                    SE T[J] = 1
                    ENCONTRAR Min em Ex[I,X] + Ds[I]
                    ENCONTRAR SMin em Ex[I,X] + Ds[I]
                    S[X] = Ex[Min,X] − Ex[SMin,X]
                            SE S[X]> S[J[Min]]
                            J[Min] = X
                            FIM SE
                    FIM SE
            FIM PARA
            PARA X de 0 até I
                    SE T[J[X]] = 1
                            PARA Y de 0 até I
                                    SE Y = X
                                    Ds[Y] = Ds[Y] + Ex[J[X],Y]
                                    Ex[J[X],Y] = 1
                                    T[X] = 0
                                    FIM SE
                                    SENÃO
                                    Ex[J[X],Y] = 0
                                    FIM SENÃO
                            FIM PARA
                    FIM SE
            FIM PARA
FIM ENQUANTO
Cmax = Ds[Max]
FIM
```

O vector *S[J]* é utilizado para guardar os valores do índice S, o vector *J[I]* guarda a tarefa com maior valor de S, que concorre pela máquina *I*. O primeiro ciclo é utilizado para calcular o índice S, de todas as tarefas que ainda não foram afectas. O vector *J[I]* deve estar preenchido com as tarefas que devem ser afectas à máquina *I* nessa iteração.

5.6.3. Implementação da heurística OMCT 3

A implementação da heurística OMCT 3, é uma versão híbrida da heurística MCT e a implementação da heurística Suffrage. O pseudo-código pode ser analisado na tabela 44.

Tabela 44 – Pseudo-Código da Heurística OMCT 3

```
INICIO
    PARA a de 0 até 1
        PARA X de 0 até J
        B = 0
        ENCONTRAR Min em Ex[I,X] + Ds[I]
        ENCONTRAR SMin em Ex[I,X] + Ds[I]
        S[X] = Ex[Min,X] − Ex[SMin,X]
            PARA Y de 0 até I
            B = B + Ex[Y,X]
            FIM PARA
        DP = √(B²/(I − 1))
        SDP[X] = a ∗ S + (1 − A) ∗ DP
        FIM PARA
Ordenar as tarefas por ordem decrescente de SDP
        PARA X de 0 até J
        ENCONTRAR Min em Ex[I,X] + Ds[I]
        Ds[Min] = Ds[Min] + Ex[Min,X]
            PARA Y de 0 até I
                SE Y=MIN
                Ex[Y,X] = 1
                FIM SE
                SENÃO
                Ex[Y,X] = 0
                FIM SENÃO
            FIM PARA
        FIM PARA
    Fim PARA
Cmax = Ds[Max]
FIM
```

Depois de calculados os valores do índice S e o desvio padrão dos tempos de processamento, as tarefas são colocadas por ordem decrescente de SDP. É de notar que a vai variar na razão de $X + 0.05$ no intervalo [0,1]. Depois das tarefas terem sido ordenadas o procedimento é semelhante ao da heurística MCT.

5.6.4. Implementação da heurística Suffrage One

A heurística Suffrage One, é de implementação semelhante ao da heurística Suffrage, apenas alocando uma tarefa por iteração. Depois de calculado o índice S, é selecionada a tarefa com o maior valor para ser afecta. O pseudo-código pode ser analisado na tabela 45.

Tabela 45 – Pseudo-Código da Heurística Suffrage One

```
INICIO
    ENQUANTO existirem tarefas por alocar
    A = 0
        PARA X de 0 até J
            SE T[J] = 1
            ENCONTRAR Min em Ex[I,X] + Ds[I]
            ENCONTRAR SMin em Ex[I,X] + Ds[I]
            S = Ex[Min,X] − Ex[SMin,X]
                SE S > A
                A = S
                J = X
                FIM SE
            FIM SE
        FIM PARA
    ENCONTRAR Min em Ex[I,J] + Ds[I]
    Ds[Min] = Ds[Min] + Ex[Min,J]
     T[J] = 0
        PARA Y de 0 até I
            SE Y=MIN
            Ex[Y,X] = 1
            FIM SE
            SENÃO
            Ex[Y,X] = 0
            FIM SENÃO
        FIM PARA
    FIM ENQUANTO
Cmax = Ds[Max]
FIM
```

Como é possível constatar o procedimento é muito semelhante ao da heurística Suffrage, com alocação semelhante ao da heurística MCT, e bastante mais simples que o procedimento da heurística OMCT 3. Durante o ciclo é calculado o índice S, das tarefas que ainda estão por alocar, para posteriormente alocar apenas a tarefa que tenha o maior índice. Esse ciclo deve ser repetido até todas as tarefas terem sido alocadas.

5.7. Conclusão

Existem várias heurísticas para resolver o problema de minimização no *makespan* em ambientes de máquinas diferentes [15] [16] [17]. No quarto capitulo foram apresentadas seis heurísticas, que alocam as tarefas através dos tempos de processamento ou das datas de conclusão, o que permitiu identificar as suas diferenças.

Neste capítulo, foram propostas quatro hipóteses de novas heurísticas para problemas em máquinas diferentes. A heurística OMCT 1, utiliza o critério de desempate da heurística Suffrage, para determinar por que sequência as tarefas são afectas. Os exemplos permitiram verificar o bom desempenho da heurística OMCT 1, em problemas onde as melhores máquinas, para cada tarefa, se encontravam distribuídas uniformemente, ou quando não existia uma tarefa que necessitasse de muito mais tempo de processamento que todas as outras e quando os tempos de execução das tarefas não são muito dispersos.

A heurística OMCT 2 utiliza uma medida de dispersão, para determinar porque ordem as tarefas eram afectas. Ao utilizar uma medida de dispersão não se assume que as tarefas serão sempre alocadas às melhores máquinas. Os exemplos permitiram verificar o bom desempenho da heurística OMCT 2, em problemas onde existiam máquinas dominantes, isto é, máquinas que são mais rápidas para a maioria das tarefas e em problemas onde os tempos de execução das tarefas são bastante dispersos.

A heurística OMCT 3, utiliza uma soma ponderada entre uma medida de dispersão e o índice utilizado na heurística Suffrage. O facto de soma ser ponderada, permite modificar a heurística para resolver problemas distintos, através da experimentação. Com a heurística OMCT 3, foi possível encontrar a solução ótima, em todos os problemas exemplo utilizados e por isso foi selecionada para o estudo computacional, no próximo capitulo.

Finalmente, a heurística Suffrage One, é semelhante a heurística Suffrage, mas afecta apenas uma tarefa por iteração. Isso não vai permitir alocar tarefas com índices S muito baixos, à máquinas a que nenhuma outra tarefa concorre. Isto torna o procedimento um pouco mais pesado, pois há necessidade que atualizar o índice sempre que uma tarefa é afecta, ao contrario da heurística Suffrage que podia afetar várias tarefas por ciclo.

No final do capítulo, foram apresentadas os algoritmos das heurísticas propostas. Os códigos da ferramenta computacional desenvolvida encontra-se comentados no anexo I.

6. ESTUDO COMPUTACIONAL

No último capítulo foram propostas e implementadas novas heurísticas para problemas de minimização do *makespan* de máquinas diferentes em paralelo.

Neste capítulo será descrito o estudo computacional realizado com o objectivo de analisar o desempenho das heurísticas de afectação propostas, quando comparadas com as heurísticas MCT [15] [16] [17], Suffrage [15] [16] [17], e ainda com as soluções ótimas disponíveis na literatura.

6.1. Introdução

Os problemas teste utilizados, foram retirados de *Sivasankaran et al* [18], que propõem uma nova heurística para problemas de máquinas diferentes em paralelo.

Os problemas académicos utilizados encontram-se no anexo II e consistem em dois grupos de vinte instâncias com dimensão variável. A dimensão das instâncias vai ser (*2X5, 2X6, ..., 2X9, 3X5, ..., 3X9, 4X5, ..., 4X9, 5X5, ..., 5X9*), representados por *Número de Máquinas X Número de Tarefas*. Foram realizados quarenta problemas, para poder assumir-se a normalidade dos dados recolhidos através do teorema do limite central, de maneira a suportar o estudo estatístico que será realizado posteriormente.

Na tabela 46, encontram-se as soluções ótimos dos problemas de teste [18], que serão usados para avaliar o desempenho das heurística implementadas.

Tabela 46 – Soluções Ótimos dos Problemas Teste [18]

	1º Grupo	2º Grupo
2X5	21	17
2X6	25	16
2X7	35	25
2X8	43	42
2X9	38	40
3X5	9	17
3X6	12	21
3X7	21	15
3X8	22	10
3X9	23	24
4X5	11	10
4X6	7	11
4X7	16	10
4X8	16	11
4X9	17	14
5X5	8	15
5X6	6	7
5X7	6	9
5X8	12	13
5X9	16	12

Os testes foram realizados num MacBook Air, com um processador 1.6GHz Intel Core 2 Duo, 4GB memória DDR3 e com o Windows 7 instalado, com todas as atualizações disponíveis [18].

6.2. Resultados Computacionais

Uma vez concluídos os testes, é possível fazer uma análise preliminar dos resultados computacionais. Os problemas resolvidos no último capítulo deixam antever um bom desempenho das heurísticas propostas.

Na tabela 47, é possível verificar os valores de *makespan*, correspondentes à afectação das tarefas por cada uma das heurísticas implementadas, para as primeiros vinte instâncias.

Tabela 47 – Resultados das Heurísticas para o 1º Grupo de Instancias

	MCT	Suffrage	OMCT 3	Suffrage One
2X5	21	21	21	21
2X6	25	25	28	25
2X7	38	35	35	35
2X8	47	43	43	43
2X9	47	38	45	38
3X5	9	12	9	12
3X6	12	15	12	15
3X7	27	27	27	27
3X8	26	23	23	23
3X9	24	25	24	25
4X5	11	11	11	11
4X6	8	7	7	7
4X7	18	16	16	16
4X8	16	18	16	18
4X9	21	18	17	19
5X5	8	8	8	10
5X6	7	7	7	7
5X7	6	6	6	6
5X8	16	13	13	17
5X9	17	20	18	20

Na tabela 48, é possível verificar os valores de *makespan* ,correspondentes à afectação das tarefas por cada uma das heurísticas implementadas, para as próximos vinte instâncias.

Tabela 48 – Resultados das Heurísticas para o 2º Grupo de Instancias

	MCT	Suffrage	OMCT 3	Suffrage One
2X5	20	17	20	17
2X6	16	19	16	19
2X7	25	25	25	25
2X8	48	42	42	42
2X9	51	40	40	40
3X5	17	18	17	18
3X6	24	21	25	24
3X7	20	20	17	20
3X8	12	12	10	12
3X9	24	24	24	24
4X5	10	10	10	10
4X6	12	11	11	11
4X7	11	11	10	11
4X8	11	11	11	11
4X9	14	14	16	19
5X5	17	15	17	17
5X6	8	8	8	8
5X7	9	9	9	9
5X8	13	19	13	19
5X9	16	15	12	14

Como é possível analisar nas tabelas 47 e 48, a heurística OMCT 3, evidencía apresentar melhor desempenho que qualquer outra heurística. A heurística OMCT 3, parece encontrar melhores soluções que a heurísticas MCT e Suffrage, havendo apenas três problemas onde uma das outras heurísticas encontra uma melhor solução. Por sua vez, a heurística Suffrage One, apresenta resultados quase sempre piores que a heurística Suffrage. Tal resultado é inesperado, pois nos problemas exemplo, o facto da Suffrage alocar mais que uma tarefa em cada iteração, pareceu poder resultar numa degradação da solução.

É de notar que a heurística proposta no artigo de onde os testes são retirados não vai ser utilizada na análise comparativa, pois apresenta características muito diferentes das heurísticas propostas. Essa heurística apresenta resultados muito próximos dos ótimos. Depois de uma solução inicial encontrada através da heurística MET, faz uma pesquisa de vizinhança para se aproximar da solução ótima do problema [18].

6.3. Análise estatística

Os resultados recolhidos serão analisados em dois aspectos: estatística descritiva e inferência estatística. No primeiro, serão calculados vários parâmetros (média, moda, mediana e desvio padrão) que melhor permitam caracterizar os resultados, também serão apresentados gráficos que permitem realizar uma análise do desempenho das várias heurísticas de uma forma mais simples. No segundo vai ser verificada a significância estatística dos resultados obtidos pelas heurísticas testadas.

O valor *makespan* encontrado por cada uma das heurísticas vai ser comparada com o solução ótima de forma a ser mais fácil interpretar dos dados recolhidos.

6.3.1. Estatística descritiva

Os resultados dos testes foram utilizados para realizar um estudo estatístico no SPSS e no Microsoft Excel. Os dados foram introduzidos como:

- Heurística utilizada;
- Grupo de Problemas;
- Dimensão do Problema;
- Diferença para a solução ótima;
- Makespan.

Nos gráficos de barras é possível analisar o desempenho de cada uma das heurísticas para cada um dos problemas, que pode ser analisar no figura 25.

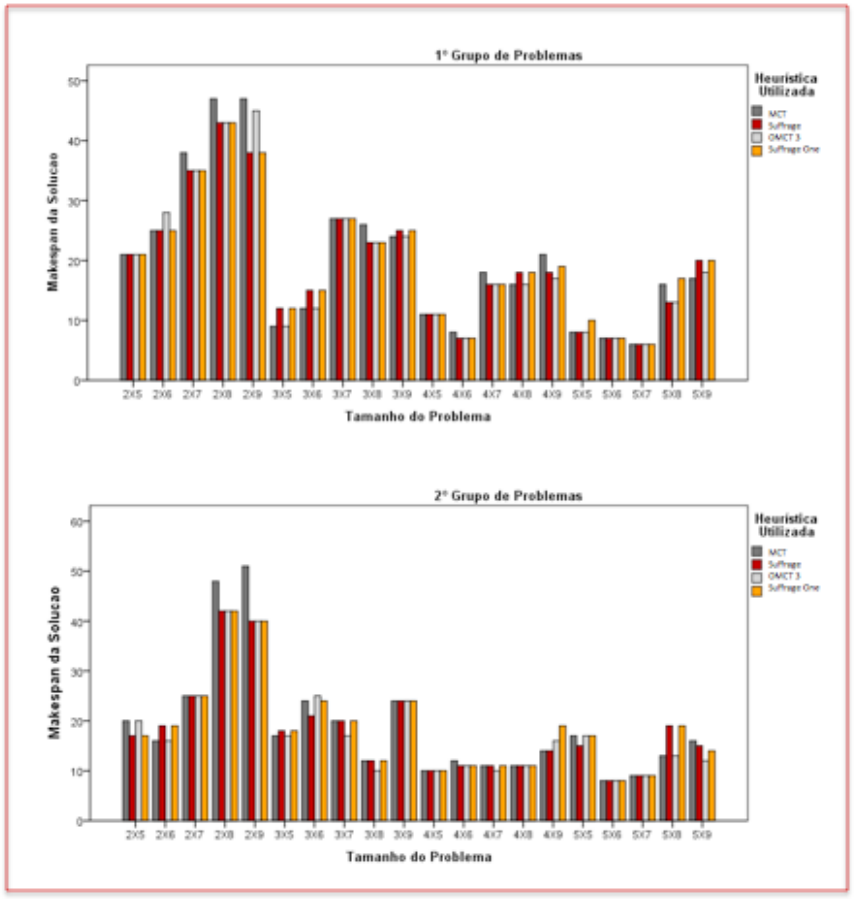

Figura 25 – Gráfico de Barras dos *Makespans* Encontrados

Como é possível analisar, não existe uma diferença significativa no desempenho de minimização do *makespan*, entre as heurísticas. A heurística OMCT 3, parece apresentar um melhor desempenho que as restantes, enquanto a heurística MCT, parece ter o pior desempenho. Isto é particularmente importante, pois a heurística OMCT 3, é uma variação da heurística MCT. Por sua vez, a heurística Suffrage One, que é uma variação da heurística Suffrage, apenas apresenta uma solução com melhor desempenho que a heurística em que se baseia, na instância *5X9*, do segundo grupo de instâncias. Isso pode indicar que a alteração realizada não surtiu os resultados esperados.

Entre as quatro heurísticas, aquelas que parecem apresentar os melhores resultados são as heurística Suffrage e a OMCT 3. Para melhor analisar as diferenças entre as heurísticas, os próximos testes não vão ter em consideração o valor de *makespan* encontrado para cada instância, mas sim o desvio em relação à solução ótima.

Utilizando o desvio em relação à solução ótima, é possível comparar os resultados de problemas com dimensões diferentes. Isso vai permitir não só ter mais dados passiveis de serem comparados, como também calcular os parâmetros de cada uma das heurísticas, uma vez que não fazia sentido calcular médias de resultados de instâncias diferentes

Como é possível analisar na tabela 49, a heurística OMCT 3, é aquela que mais vezes encontrou a solução ótima do problema, em 27 vezes, enquanto a heurística MCT, apenas encontrou a solução ótima em 17 problemas. Por sua vez a heurística Suffrage, encontrou mais vezes a solução ótima, em 22 problemas, que a heurística Suffrage One.

Tabela 49 – Tabela de Frequência da Diferença para a Solução Ótima

Diferença para *Makespan* Ótimo	Heurística Utilizada				Total
	MCT	Suffrage	3º Hipótese	4º Hipótese	
0	17	22	26	18	83
1	7	7	5	5	24
2	3	3	3	7	16
3	3	4	3	4	14
4	5	1	1	1	8
5	1	1	0	3	5
6	2	2	1	2	7
7	0	0	1	0	1
9	1	0	0	0	1
11	1	0	0	0	1
Total	40	40	40	40	160

Uma análise mais cuidada, permite concluir que todas as heurísticas apresentaram resultados próximos do ótimo. Em mais de 50% dos casos, as heurísticas analisadas encontram a solução ótima do problema de minimização do *makespan*.

A heurística MCT, é aquela que apresenta os dois piores resultados, com uma diferença para solução ótima de 9 e 11. Já a heurística OMCT 3, que parece obter os melhores resultados, foi a que encontrou a solução com o 3º maior desvio em relação a solução ótima. A Suffrage e a Suffrage One, parecem ser as duas heurísticas que apresentam menor dispersão, não obtendo nenhum solução com desvio da solução ótima superior a 6.

Para facilitar a interpretação dos resultados foram, também, feitos gráficos de barras da diferença dos resultados de cada uma das heurísticas, em relação a solução ótima, que podem ser analisados na figura 26.

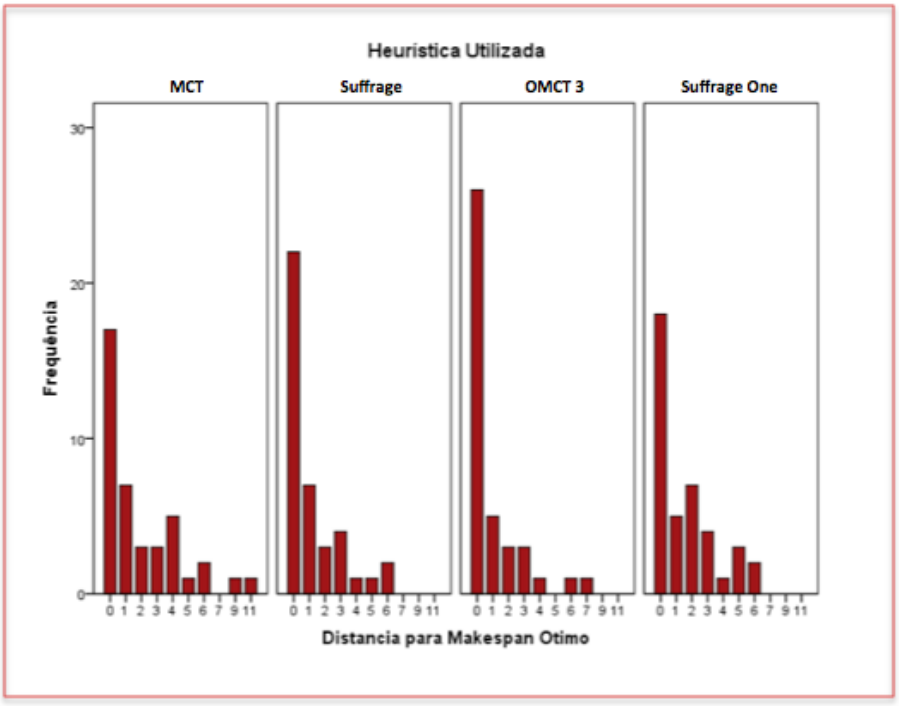

Figura 26 – Gráfico de Barras da Desvio em Relação a Solução Ótima

Para analisar a dispersão, identificar os *outliers* e verificar amplitude interquartil, são apresentados, na figura 27, os *boxplots* de cada uma das heurísticas.

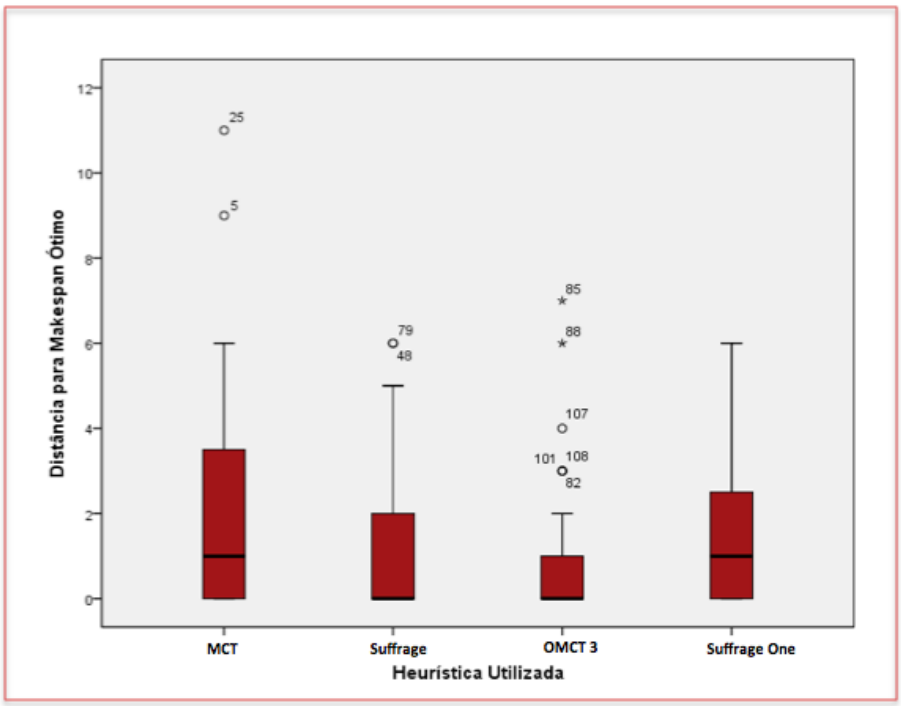

Figura 27 – *Boxplot* da Desvio em Relação a Solução Ótima

É possível concluir, que todas as heurísticas apresentam distribuições assimétricas positivas, quando comparadas com solução a ótima. Existem *outliers* para todas as heurísticas, sendo que para a heurística OMCT 3, existem dois *outliers* severos. Os *outliers* ou valores extremos, não podem ser explicados por erros na introdução dos dados, sendo resultado da natureza incerta das heurísticas. Tendo em consideração esses valores, a heurística OMCT 3, é aquela que apresenta a menor amplitude interquartil.

Finalmente, é possível calcular os parâmetros para cada uma das heurísticas. Na tabela 50, é possível analisar a média, mediana, moda e desvio padrão de cada uma das heurísticas.

Tabela 50 – Parâmetros de cada Heurística

	MCT	Suffrage	OMCT 3	Suffrage One
Média	1.98	1.15	0.93	1.55
Mediana	1	0	0	1
Moda	0	0	0	0
Desvio Padrão	2.626	1.718	1.685	1.867

Assim é possível afirmar que nos testes realizados, a alteração feita à heurística MCT, apresentou melhores resultados que a heurística em que se baseia. Em média tem um desvio do resultado ótimo de 0.93, que é inferior ao de qualquer outra heurística. O desvio padrão da heurística OMCT 3, também é o mais pequeno de todas as heurísticas. Por sua vez, a heurística Suffrage One, evidencia apresentar um desempenho inferior, que o da heurística Suffrage.

Foram ainda calculados os desvios médios, entre os *makespans* calculados nas instâncias de testes, pela heurística MCT e a OMCT 3, e a heurística Suffrage e a Suffrage One. Os desvios foram calculados através da expressão:

$$\frac{\sum \frac{|Makespan\ Hipótese - Makespan\ Heurística|}{Makespan\ Heurística}}{40} \qquad (26)$$

Existe uma diferença entre os resultados da heurística MTC e da OMCT 3, da ordem dos 5.95%. A diferença entre os resultados da heurística Suffrage e da Suffrage One, é menos significativa, sendo da ordem dos 3.28%.

No próxima secção será analisado se as diferenças entre as heurísticas, são estatisticamente significativas [18] [19].

6.3.2. Inferência Estatística

A inferência estatística, é o ramo da estatística que procura tirar conclusões sobre a população, com base numa amostra. Devem ser sempre acompanhadas pelo grau de confiança que se tem nessas conclusões [19].

Para comparar as várias heurísticas, vai ser utilizado o teste ANOVA, com dois factores. Isso vai permitir comparar como o desvio do resultado ótimo é afectado pela heurística escolhida e pela dimensão do problema. Finalmente, vai ainda ser verificado se existe uma relação entre os dois factores analisados, isto é, se existe uma relação entre o desempenho da heurística utilizada e a dimensão do problema. Os testes realizados, foram:

1º Factor, Heurística:

- Ho: Não existe uma diferença significativa de desempenho entre as várias heurísticas, no que diz respeito ao desvio ao *makespan* ótimo.
- H1: Existe uma diferença significativa de desempenho entre as várias heurísticas, no que diz respeito ao desvio ao *makespan* ótimo.

2º Factor, Dimensão:

- Ho: Não existe uma diferença significativade de desempenho entre problemas de dimensões diferentes, no que diz respeito ao desvio ao *makespan* ótimo.
- H1: Existe uma diferença significativa de desempenho entre entre problemas de dimensões diferentes, no que diz respeito ao desvio ao *makespan* ótimo.

Interação entre factores:

- Ho: Não existe relação entre a heurística utilizada e a dimensão do problema, no que diz respeito ao desvio ao *makespan* ótimo.
- H1: Existe relação entre a heurística utilizada e a dimensão do problema, no que diz respeito ao desvio ao *makespan* ótimo.

Os resultados dos testes ANOVA, com dois factores, podem ser vistos na tabela 51.

Tabela 51 – Teste ANOVA

Source	*Type III Sum of Squares*	*df*	*Mean Square*	*F*	*Sig.*
Corrected Model	510,400[a]	79	6,461	3,540	,000
Intercept	313,600	1	313,600	171,836	,000
Heurística	25,650	3	8,550	4,685	,005
Dimensão	254,150	19	13,376	7,329	,000
Heurística * Dimensão	230,600	57	4,046	2,217	,001
Error	146,000	80	1,825		
Total	970,000	160			
Corrected Total	656,400	159			

Com um nível de significância de 5%, não é possível aceitar a hipótese nula, com um *p-value* de 0,005. Assim, pode-se assumir que pelo menos uma heurística é significativamente, diferente de uma das outras. Para verificar onde existem as diferenças de desempenho, foi realizado um teste *Post Hoc*, teste de Scheffe, cujos resultados podem ser analisados na tabela 52.

Tabela 52 – Teste de Scheffe

(I) Heurística Utilizada	(J) Heurística Utilizada	*Mean Difference (I-J)*	*Std. Error*	*Sig.*
MCT	Suffrage	,82	,302	,067
	OMCT 3	1,05*	,302	,010
	Suffrage One	,42	,302	,579
Suffrage	MCT	-,82	,302	,067
	OMCT 3	,23	,302	,906
	Suffrage One	-,40	,302	,627
	MCT	-1,05*	,302	,010
OMCT 3	Suffrage	-,23	,302	,906
	Suffrage One	-,63	,302	,241
	MCT	-,42	,302	,579
Suffrage One	Suffrage	,40	,302	,627
	OMCT 3	,63	,302	,241

É possível ver que apenas existe uma diferença entre a heurística MCT e a OMCT 3, *p-value* de 0,005. Como a heurística OMCT 3, apresenta uma menor distância média ao *makespan* ótimo, que a heurística MCT, existem evidências estatísticas de que a OMCT 3, tem melhor desempenho que a heurística MCT. Da mesma forma, não é possível concluir que a heurística Suffrage, apresente resultados melhores que Suffrage One, embora os resultados amostrais da primeira tenha sido mais próximos do resultado ótimo.

Do mesmo modo, também não é possível aceitar a hipótese nula para o factor dimensão, *p-value* de 0,000. Isto quer dizer que a distância ao *makespan* ótimo, é afectada pela dimensão do problema. Tal não é de estranhar, tendo em conta que os problemas apresentam características diferentes. Para este fatores, não foram feitos os testes *Post Hoc*, porque existem vinte instâncias com dimensões diferentes, bem como, devido ao facto da própria natureza dos problemas não permitir retirar conclusões destes testes.

Finalmente, também não é possível afirmar que o desempenho de uma heurística é independente da dimensão do problema, *p-value* de 0,001 [18] [19].

6.4. Conclusão

Os resultados das heurísticas propostas foram bastante diferentes, enquanto a heurística OMCT 3, pareceu apresentar resultados mais próximos do ótimo que a heurística MCT, a heurística Suffrage One, não apresenta vantagens quando comparada com a Suffrage.

A diferença de resultados entre a heurística OMCT 3 e a heurística MCT, nos problemas teste, foram da ordem dos 5.95%, enquanto que a diferença entre a heurística Suffrage One e a heurística Suffrage, foram da ordem dos 3.28%. A heurística OMCT 3, foi ainda aquela que encontrou os resultados mais próximos do resultado ótimo, com um desvio médio em relação a solução ótima de 0.93, inferior ao de qualquer uma das outra heurística.

Finalmente foi utilizado o teste ANOVA, para verificar se os resultados amostrais podem indiciar diferenças no desempenho das heurísticas. O resultado do teste demostrou que não existem diferenças significativas, entre a heurística Suffrage One e a heurística Suffrage, mas que existem diferenças significativas, entre a heurística OMCT 3 e a heurística MCT. Isso deixa entender que a OMCT 3, apresenta melhor desempenho que a heurística MCT, onde as tarefas são alocadas por ordem aleatória.

Foi ainda possível verificar que a qualidade das soluções de cada uma das heurísticas não é independente da dimensão do problema, o que se deve à natureza das própria heurísticas. A heurística MCT, ao afectar as tarefas numa ordem aleatória deverá ter pior desempenho em problemas de maiores dimensões. A heurística Suffrage, deverá ter pior desempenho em problemas onde o número de máquinas seja próximo do número de tarefas, ao alocar mais que uma tarefa por iteração.

7. CONCLUSÃO

Com o aumento da complexidade e com a necessidade de flexibilização dos sistemas produtivos, tornou-se impossível abordar o problema do escalonamento, como era abordado durante a revolução industrial. Atualmente, o escalonamento da produção é um campo científico bastante estudado, particularmente na área das Ciências de Computação, muito devido à necessidade de utilizar meios informáticos para resolver os problemas mais complexos.

Continuam a ser desenvolvidas novas heurísticas capazes de produzir melhores resultados para problemas de escalonamento, impossíveis de resolver em tempo útil através de métodos exatos. Neste trabalho foi abordado o problema da minimização do *makespan* em ambientes de máquinas diferentes, em paralelo ($R_m||C_{max}$), que não pode ser descrito por uma função de complexidade polinomial e por isso não pode ser resolvido de forma eficiente através métodos enumerativos. Este problema tanto pode descrever a afectação num ambiente de produção, quando as máquinas não tem o mesmo desempenho, como num sistema de computação partilhada, onde os computadores, habitualmente, não tem todos as mesmas características de processamento, armazenamento, etc. É importante notar que em muitas unidades produtivas existem, de facto, máquinas com desempenhos diferentes, dispostas em paralelo, enquanto a computação partilhada é cada vez mais utilizada, especialmente por universidades ou laboratórios.

7.1. Objectivos

Os problemas de escalonamento têm sido tradicionalmente abordados como problemas de optimização sujeitos a restrições, cujos elementos básicos são as máquinas e as tarefas. Pode considerar-se que o escalonamento é constituído por duas fases: a afectação ou atribuição das operações das tarefas aos recursos do sistema e o respectivo sequenciamento e calendarização. Estas fases podem ser tratadas separadamente ou duma forma integrada, dependendo do tipo e dimensão do sistema em questão e da estratégia de escalonamento.

Pretendeu-se desenvolver duas heurísticas para a afectação de tarefas aos recursos, em sistemas de máquinas diferentes em paralelo.

7.2. Trabalho realizado

Uma análise de seis das heurísticas mais utilizadas para resolver este tipo de problemas permitiu identificar as que utilizam os tempos de processamento e as que utilizam as datas de conclusão, bem como, aquelas que utilizam uma versão híbrida considerando tanto os tempos de processamento como as datas de conclusão. Ao utilizar os tempos de processamento, toda afectação pode ser feita numa única matriz, isto é, não é preciso atulizar as disponibilidades das máquinas após uma tarefa ter sido alocada, o que pode levar à alocação de muitas tarefas à mesma máquina. Utilizando as datas de conclusão, é necessário atualizar as disponibilidades das máquinas sempre que uma tarefa é alocada, o que aumentar o esforço necessário para encontrar a solução.

Neste trabalho foram estudadas quatros modificações a duas das heurísticas analisadas, MCT e Suffrage, nas quais foram identificadas limitações e oportunidades de melhoria.

A heurística MCT, utiliza as datas de conclusão para afectar as tarefas numa ordem aleatória, o que pode resultar em soluções muito díspares. Propõem-se, no trabalho, três modificações à heurística determinando uma ordem para a alocação das tarefas. A heurística OMCT 1, ordena as tarefas através do índice S, semelhante ao índice utilizado na heurística Suffrage, o que não resultou em soluções satisfatória, uma vez que índice S apenas tinha em consideração as duas melhores máquinas. A heurística Suffrage [15] [16] [17], ultrapassa este problema recalculando o índice após a afectação de cada tarefa, o que não era possível na heurística MCT [15] [16] [17].

A heurística OMCT 2, ordena as tarefas através do desvio padrão dos tempos de processamento nas várias máquinas, isto é, foi utilizada uma medida de dispersão para ultrapassar a necessidade de alterar a ordem de afectação, depois de alocada cada uma das tarefas. No entanto ao utilizar apenas uma medida de dispersão, todas as máquinas têm o mesmo peso na determinação da ordem de afectação das tarefas.

A heurística OMCT 3, utiliza a soma ponderada entre o índice S e o desvio padrão dos tempos de processamento, para decidir qual a sequência em que as tarefas vão ser alocadas. Este procedimento vai atribuir prioridade às tarefas que tenham uma maior dispersão nos tempos de processamento e, ao mesmo tempo, atribuir maior importância às duas máquinas mais rápidas.

A heurística Suffrage, pode alocar várias tarefas de uma única vez, com cada uma a concorrer pela melhor máquina. O facto de poderem ser alocadas várias tarefas numa única iteração resultou em piores soluções no problema teste, sendo por isso analisada, na heurística Suffrage One, uma modificação a heurística, onde apenas uma tarefa é alocada por iteração.

Foram implementadas quatro heurísticas: as duas heurísticas não modificadas e duas das modificações propostas. As heurísticas OMCT 1 e OMCT 2, não foram implementadas pois é possível analisar os resultados de ambas na heurística OMCT 3.

Para analisar o desempenho de cada uma das modificações foram utilizados quarenta instâncias retirados de *Sivasankaran et al* [18]. Os testes computacionais demostraram que modificação à heurística MCT, resultou em soluções melhores que as da heurística original, com um desvio médio de 5.95%. A modificação à heurística Suffrage One, não resultou em soluções com diferença estatisticamente significativa em relação as soluções encontradas pela heurística não modificada.

Em ambos os casos, é possível afirmar que as heurísticas modificadas são mais pesadas em tempos computacionais, do que as heurísticas não modificadas. Isto deve-se ao facto de serem necessários cálculos adicionais para determinar a ordem em que as tarefas vão ser alocadas na modificação da heurística MCT, e ainda ao facto da modificação à heurística Suffrage, apenas alocar uma única tarefa em cada iteração.

7.3. Perspetivas de trabalho futuro

Como trabalho futuro, propõem-se a realização de um estudo computacional mais extenso, particularmente em problemas de maiores dimensões. Esse estudo pode explicar o mau desempenho da heurística Suffrage One, depois de identificada a fraqueza da alocação de várias tarefas numa única iteração, no problema teste da heurística Suffrage. Um estudo computacional mais extenso, poderá permitir analisar como a modificação da heurística MCT, se comporta, se forem impostos limites na pesquisa da soma ponderada entre o índice S e o desvio padrão.

Pensa-se que também poderia ser útil analisar, como cada uma das heurísticas modificadas se comportam no que diz respeito à disponibilidades das máquinas de não *makespan*. Isto poderá levar à determinação de uma heurística, mesmo quando todas apresentam um *makespan* estatisticamente semelhante.

Referências Bibliográficas

[1] DILWORTH, James B. – Production and Operations Management; Fifth Edition; McGraw-Hill International Editions.

[2] KUMAR, S. Anil; SURESH, N. – Production and Operations Management (With Skill Development, Caselets and Cases); Second Edition; New Age International Publishers.

[3] SLACK, Nigel; CHAMBERS, Stuart; JOHNSTON, Robert – Operations Management; Sixth Edition; Prentice Hall.

[4] REID, R. Dan, SANDERS, Nada R. – Operations Management An Integrated Approach; Fourth Edition; John Wiley & Sons, Inc.

[5] COURTOIS, A. ; PILLET, M; MARTIN, C. – Gestão da Produção; Quarta Edição; Lidel.

[6] PINEDO, Michael L. – Scheduling Theory, Algorithms, and Systems; Forth Edition; Springer, 2010.

[7] LOPEZ, Pierre; ROUBELLAT, François – Production Scheduling; CAM.

[8] HERRMAN, W. Jeffrey – Handbook of Production Scheduling; Springer.

[9] BLAZEWICZ, J. ; ECKER, K. H. ; PESH E. ; SCHMIDT, G. ; WEGLARZ, J. – Scheduling Computer and Manufacturing Processes; Springer.

[10] BAKER, K. ; TRIETSCH, D. – Principles of Sequencing and Scheduling; Wiley.

[11] BRUCKER, Peter – Scheduling Algorithms; Fifth Edition; Springer.

[12] PEREIRA, Ana M. – Aplicação de Meta-Heurísticas ao Problema de Escalonamento em Ambiente Dinâmico de Produção Discreta; Tese de Doutoramento; Universidade do Minho, Escola de Engenharia, Departamento de Produção e Sistemas.

[13] PFUND, Michele; FOWLER, John; GUPTA, Jatinder – A Survey of Algorithms for Single and Multi-Objective Unrelated Parallel-Machine Deterministic Scheduling Problems; Journal of the Chinese Institute of the Industrial Engineers, Vol. 23, No 3.

[14] FIBICH, Pavel; MATYSKA, Ludek; RUDOVÁ, Hana – Model of Grid Scheduling Problem, Faculty of Informatics, Masaryk University.

[15] GUPTA, Kamali; SINGH, Manpreet – Heuristic Based Task Scheduling In Grid, International Journal of Engineering and Technology.

[16] ETMINANI, Kobra; NAGHIBAZADEH, Mahmoud; YANEHSARI, Noorali R. – A Hybrid Min-Min Max-Min Algorithm with Improved Performance, Department of Computer Engineering, University of Mashad; Department of IT, Iran Khodro Khorasan.

[17] BRICEÑO, Luis Diego; SIEGEL, Howard Jay, OLTIKAR, Mohana; MACIEJEWSKI, Anthony A. – Characterization of the Iterative Application of Makespan Heuristics on Non-Makespan Machines in a Heterogeneous Parallel and Distributed Environment, Springer Science+Business Media

[18] SIVASANKARAN, P; SORNAKUMAR, T; PANNEERSELVAM, R – Efficient Heuristic to Minimize Makespan in Single Machine Problem with Unrelated Parallel Machines. Department of Management Studies, School of Management, Pondicherry University, Pondicherry, India. Department of Mechanical Engineering, Thiagarajar College of Engineering, Madura, India.

[19] FIELD, Andy – Discovering Statistics Using SPSS, Third Edition, Sage Publications.

Anexos

Anexo I

MCT

```
#include <stdio.h>
#include<time.h>
int main ()
{
int maq, tar; // Número de máquinas e tarefas
float Ex[100][100]; // Matriz com os tempos de execução
float Ds[100]; // Vector com as disponibilidades das máquinas
int x,y,z;
float aux;
int Min;
int Max;
float Soma;
float temp;
double dur;
FILE *af;
clock_t start,stop;
printf ("Numero de maquinas:");
scanf ("%d", &maq);
printf ("\nNumero de tarefas:");
scanf ("%d", &tar);
for (x=0;x<tar;x++)
        {
        for (y=0;y<maq;y++)
                {
                printf ("\nTempo de execucao da tarefa %d na maquina %d:", x+1, y+1);
                scanf ("%f", &Ex[y][x]);
                }
        }
for (x=0;x<maq;x++)
        {
        printf ("\nDisponibilidade inicial da maquina %d:", x+1);
        scanf ("%f", &Ds[x]);
        }
```

```
start = clock(); // Iniciar o clock
for (x=0;x<tar;x++)
        {
        aux=10000;
        for (y=0;y<maq;y++) // Entre máquina com a menor data de conclusão para a tarefa
                {
                Soma=Ds[y]+Ex[y][x];
                if (Soma<aux)
                        {
                        aux=Soma;
                        Min=y;
                        }
                }
        Ds[Min]=Ds[Min]+Ex[Min][x]; // Atualiza a disponibilidade da máquina
        for (y=0;y<maq;y++) // Atualiza a matriz Ex para mostrar a alocação
                {
                Ex[y][x]=0;
                }
        Ex[Min][x]=1;
        }
stop=clock(); // Para o clock
dur=(double)stop-start;
printf ("\nAlocacao");
for (x=0;x<tar;x++)
        {
        printf ("\n");
        for (y=0;y<maq;y++)
                {
                printf ("%f ", Ex[y][x]); // Imprime a alocação
                }
        }
printf ("\n\nDisponibilidades\n");
for (y=0;y<maq;y++)
        {
        printf ("%f ", Ds[y]); // Imprime as disponibilidades
        }
aux=-1;
for (y=0;y<maq;y++)
```

```
                {
                temp=Ds[y];
                if (temp>aux)
                        {
                        aux=temp; // Encontra o makespan
                        }
                }
printf ("\n\nMakespan\n");
printf ("%f", aux); // Imprime o makespan
printf ("\n\nDuracao\n");
printf("%f",(dur/CLOCKS_PER_SEC)*1000); // Imprime o tempo
af=fopen("MCT.txt", "w");
fprintf (af, "\nAlocacao");
for (x=0;x<tar;x++)
        {
        fprintf (af, "\n");
        for (y=0;y<maq;y++)
                {
                fprintf (af, "%f ", Ex[y][x]); // Imprime a alocação
                }
        }
fprintf (af, "\n\nDisponibilidades\n");
for (y=0;y<maq;y++)
        {
        fprintf (af, "%f ", Ds[y]); // Imprime as disponibilidades
        }
aux=-1;
for (y=0;y<maq;y++)
                {
                temp=Ds[y];
                if (temp>aux)
                        {
                        aux=temp; // Encontra o makespan
                        }
                }
fprintf (af, "\n\nMakespan\n");
fprintf (af, "%f", aux); // Imprime o makespan
fprintf (af, "\n\nDuracao\n");
fprintf(af, "%f",(dur/CLOCKS_PER_SEC)*1000); // Imprime o tempo
```

```
fclose(af);
getchar();
getchar();
return 0;
}
```

Suffrage

```
#include <stdio.h>
#include<time.h>
int main ()
{
int maq, tar; ; // Número de máquinas e tarefas
float Ex[100][100]; // Matriz com os tempos de execução
float Ds[100]; // Vector com as disponibilidades das máquinas
float S[100]; // Vector com o índice S
int J[100]; // Vector com as tarefas que vão ser alocadas a cada maquina
int T[100]; // Vector que mostra as tarefas alocadas
int x,y,z;
int f;
float aux;
float Soma;
float temp;
int Min;
int SMin;
double dur;
FILE *af;
clock_t start,stop;
printf ("Numero de maquinas:");
scanf ("%d", &maq);
printf ("\nNumero de tarefas:");
scanf ("%d", &tar);
for (x=0;x<tar;x++)
        {
        T[x]=1; // Nenhuma tarefa foi alocada
        }
for (x=0;x<tar;x++)
        {
        for (y=0;y<maq;y++)
                {
                printf ("\nTempo de execucao da tarefa %d na maquina %d:", x+1, y+1);
                scanf ("%f", &Ex[y][x]);
                }
        }
```

```
for (x=0;x<maq;x++)
        {
        printf ("\nDisponibilidade inicial da maquina %d:", x+1);
        scanf ("%f", &Ds[x]);
        }
f=tar; // Número de tarefas para alocar

start=clock(); // Inicia clock
while (f>0) // Repetir até todas as tarefas terem sido alocadas
        {
        Min=0;
        SMin=0;
        for (x=0;x<tar;x++)
                {
                S[x]=-1; // Índice S iniciado -1
                }
        for (x=0;x<maq;x++)
                {
                J[x]=tar; // Iniciado tar para evitar alocações
                }
        for (x=0;x<tar;x++)
                {
                if (T[x]==1) // Apenas para as tarefas por alocar
                        {
                        aux=10000;

                        for (y=0;y<maq;y++) // Encontrar a menor data de conclusão
                                {
                                Soma=Ds[y]+Ex[y][x];
                                if (Soma<aux)
                                        {
                                        aux=Soma;
                                        Min=y;
                                        }
                                }
                        aux=10000;
                        for (z=0;z<maq;z++) // Encontrar a 2° menor data de conclusão
                                {
                                if (z!=Min)
```

```
                                        {
                                        Soma=Ds[z]+Ex[z][x];
                                        if (Soma<aux)
                                                {
                                                aux=Soma;
                                                SMin=z;
                                                }
                                        }
                                }
                        S[x]=(Ex[SMin][x]+Ds[SMin])-(Ex[Min][x]+Ds[Min]); // Índice S
                        if (S[x]>S[J[Min]]) // Escolha da tarefa a alocar em cada máquina
                                {
                                J[Min]=x;
                                }
                        }
                }
for (x=0;x<maq;x++)
                {
                if (T[J[x]]==1)
                        {
                        for (y=0;y<maq;y++)
                                {
                                if (y==x)
                                        {
                                        Ds[y]=Ds[y]+Ex[y][J[x]]; // Disponibilidades
                                        Ex[y][J[x]]=1; // Alocação
                                        T[J[x]]=0; // Já foi alocada
                                        f--; // Menos uma tarefa por alocar
                                        }
                                else
                                        {
                                        Ex[y][J[x]]=0; // Alocação
                                        }
                                }
                        }
                }
        }
stop=clock(); // Para clock
dur=(double)stop-start;
```

```
printf ("\nAlocacao");
for (x=0;x<tar;x++)
	{
	printf ("\n");
	for (y=0;y<maq;y++)
		{
		printf ("%f ", Ex[y][x]); // Imprime alocação
		}
	}
printf ("\n\nDisponibilidades\n");
for (y=0;y<maq;y++)
	{
	printf ("%f ", Ds[y]); // Imprime disponibilidades
	}
aux=-1;
for (y=0;y<maq;y++)
		{
		temp=Ds[y];
		if (temp>aux)
			{
			aux=temp; // Encontra o makespan
			}
		}
printf ("\n\nMakespan\n");
printf ("%f", aux); // Imprime makespan
printf ("\n\nDuracao\n");
printf("%2f",(dur/CLOCKS_PER_SEC)*1000); // Imprime tempo
af=fopen("Suffrage.txt", "w");
fprintf (af, "\nAlocacao");
for (x=0;x<tar;x++)
	{
	fprintf (af, "\n");
	for (y=0;y<maq;y++)
		{
		fprintf (af, "%f ", Ex[y][x]); // Imprime alocação
		}
	}
fprintf (af, "\n\nDisponibilidades\n");
for (y=0;y<maq;y++)
```

```
        {
        fprintf (af, "%f ", Ds[y]); // Imprime disponibilidade
        }
aux=-1;
for (y=0;y<maq;y++)
                {
                temp=Ds[y];
                if (temp>aux)
                        {
                        aux=temp; // Encontra makespan
                        }
                }
fprintf (af, "\n\nMakespan\n");
fprintf (af, "%f", aux); // Imprime makespan
fprintf (af, "\n\nDuracao\n");
fprintf(af, "%f",(dur/CLOCKS_PER_SEC)*1000); // Imprime tempo
fclose(af);
getchar();
getchar();
return 0;
}
```

OMCT 3

```
#include <stdio.h>
#include <math.h>
#include<time.h>
int main ()
{
int maq, tar; // Número de máquinas e tarefas
int Min, SMin;
int x, y, iter;
float aux;
float Soma, Mean; // Auxiliares para o cálculo do desvio padrão
float b, d;
float temp;
double a, dur;
int Or[100]; // Ordem das tarefas para alocação
float Ex[100][100], Ex1 [100][100]; // Matriz com tempos de execução e alocação
float Ds[100], Ds1[100]; // Vector com as disponibilidades
double SDP[100], S[100], DP[100]; // Índice S, desvio padrão e some ponderada
FILE *af;
clock_t start,stop;
printf ("Numero de maquinas:");
scanf ("%d", &maq);
printf ("\nNumero de tarefas:");
scanf ("%d", &tar);
for (x=0;x<tar;x++)
        {
        for (y=0;y<maq;y++)
                {
                printf ("\nTempo de execucao da tarefa %d na maquina %d:", x+1, y+1);
                scanf ("%f", &Ex[y][x]);
                }
        }
for (x=0;x<maq;x++)
        {
        printf ("\nDisponibilidade inicial da maquina %d:", x+1);
        scanf ("%f", &Ds[x]);
        }
```

```
start=clock(); // Inicia clock
for (x=0;x<tar;x++)
        {
        Min=0;
        SMin=0;
        aux=10000;
        for (y=0;y<maq;y++)
                {
                Soma=Ds1[y]+Ex[y][x];
                if (Soma<aux) // Encontra menor data de conclusão
                        {
                        aux=Soma;
                        Min=y;
                        }
                }
        aux=10000;
        for (y=0;y<maq;y++)
                {
                if (y!=Min)
                        {
                        Soma=Ds1[y]+Ex[y][x];
                        if (Soma<aux) // Encontra 2° menor data de conclusão
                                {
                                aux=Soma;
                                SMin=y;
                                }
                        }
                }
        S[x]=(Ex[SMin][x]+Ds1[SMin])-(Ex[Min][x]+Ds1[Min]); // Índice S
        b=0;
        for (y=0;y<maq;y++)
                {
                b=b+Ex[y][x];
                }
        Mean=b/maq;
        d=0;
        for (y=0;y<maq;y++)
                {
                b=pow(Ex[y][x]-Mean,2);
```

```
                d=b+d;
                }
        DP[x]=sqrt(d/maq); // Desvio padrão
        }
af=fopen("3Hipotese.txt", "w");
iter=1;
a=0; // Variação da soma ponderada
while (iter<=21) // 21 iterações de 0 a 1
        {
        for (x=0;x<maq;x++)
                {
                Ds1[x]=Ds[x];
                }
        for (x=0; x<tar; x++)
                {
                SDP[x]=a*S[x]+(1-a)*DP[x]; // Cálculo da some ponderada
                }
        for (x=0;x<tar;x++)
                {
                aux=-1;
                for (y=0;y<tar;y++) // Determina ordem da alocação
                        {
                        if (SDP[y]>aux)
                                {
                                Or[x]=y;
                                aux=SDP[y];
                                }
                        }
                SDP[Or[x]]=-1;
                }
        for (x=0;x<tar;x++)
                {
                aux=10000;
                for (y=0;y<maq;y++)
                        {
                        Soma=Ds1[y]+Ex[y][Or[x]];
                        if (Soma<aux) // Encontra a menor data de conclusão
```

```
                        {
                        aux=Soma;
                        Min=y;
                        }
                }
        Ds1[Min]=Ds1[Min]+Ex[Min][Or[x]];
        for (y=0;y<maq;y++)
                {
                Ex1[y][Or[x]]=0; // Alocação
                }
        Ex1[Min][Or[x]]=1;
        }

stop=clock(); // Para clock
dur=(double)stop-start;
printf ("\n\n\n\n%d Iteracao", iter); // Imprime iteração
printf ("\n\nAlocacao");
for (x=0;x<tar;x++)
        {
        printf ("\n");
        for (y=0;y<maq;y++)
                {
                printf ("%f ", Ex1[y][x]); // Imprime alocação
                }
        }
printf ("\n\nDisponibilidades\n");
for (y=0;y<maq;y++)
        {
        printf ("%f ", Ds1[y]); // Imprime disponibilidade
        }
aux=-1;
for (y=0;y<maq;y++)
        {
        temp=Ds1[y];
        if (temp>aux)
                {
                aux=temp; // Encontra makespan
                }
        }
```

```
        printf ("\n\nMakespan\n");
        printf ("%f", aux); // Imprime makepsan
        printf ("\n\na\n");
        printf ("%f", a); // Imprime a
        fprintf (af, "\n\n\n\n%d Iteracao", iter); // Imprime iteração
        fprintf (af, "\n\nAlocacao");
        for (x=0;x<tar;x++)
                {
                fprintf (af, "\n");
                for (y=0;y<maq;y++)
                        {
                        fprintf (af, "%f ", Ex1[y][x]); // Imprime alocação
                        }
                }
        fprintf (af, "\n\nDisponibilidades\n");
        for (y=0;y<maq;y++)
                {
                fprintf (af, "%f ", Ds1[y]); // Imprime disponibilidade
                }
        aux=-1;
        for (y=0;y<maq;y++)
                        {
                        temp=Ds1[y];
                        if (temp>aux)
                                {
                                aux=temp; // Encontra makespan
                                }
                        }
        fprintf (af, "\n\nMakespan\n");
        fprintf (af, "%f", aux); // Imprime makespan
        fprintf (af, "\n\na\n");
        fprintf (af, "%f", a); // Imprime a
        a=a+0.05; // Atualiza a
        iter++; // Atualiza a iteração
        }
printf ("\n\n\n\nDuracao Total\n");
printf("%f",(dur/CLOCKS_PER_SEC)*1000); // Imprime tempo
fprintf (af, "\n\n\n\nDuracao Total\n");
fprintf(af, "%f",(dur/CLOCKS_PER_SEC)*1000); // Imprime tempo
```

```
fclose(af);
getchar();
getchar();
return 0;
}
```

Suffrage One

```
#include <stdio.h>
#include<time.h>
int main ()
{
int T[100]; // Vector com as tarefas por alocar
float Ex[100][100]; // Matriz com os tempos de execução
float Ds[100]; // Vector com as disponibilidades das máquinas
float S[100]; // Vector com o índice S
int maq, tar; // Número de máquinas e tarefas
int x,y,z;
int J, F;
int Min, SMin;
float aux, aux1;
float temp, Soma;
double dur;
FILE *af;
clock_t start,stop;
printf ("Numero de maquinas:");
scanf ("%d", &maq);
printf ("\nNumero de tarefas:");
scanf ("%d", &tar);
for (x=0;x<tar;x++)
        {
        T[x]=1; // Nenhuma tarefa alocada
        }
for (x=0;x<tar;x++)
        {
        for (y=0;y<maq;y++)
                {
                printf ("\nTempo de execucao da tarefa %d na maquina %d:", x+1, y+1);
                scanf ("%f", &Ex[y][x]);
                }
        }
for (x=0;x<maq;x++)
        {
        printf ("\nDisponibilidade inicial da maquina %d:", x+1);
        scanf ("%f", &Ds[x]);
```

```
        }
F=tar;
start=clock(); // Inicia clock
while (F>0) // Repetir até todas as tarefas terem sido alocadas
        {
        Min=0;
        SMin=0;
        aux1=-1;
        for (x=0;x<tar;x++)
                {
                if (T[x]==1) // Apenas para as tarefas ainda não terem sido alocadas
                        {
                        aux=10000;
                        for (y=0;y<maq;y++)
                                {
                                Soma=Ds[y]+Ex[y][x];
                                if (Soma<aux) // Menor data de conclusão
                                        {
                                        aux=Soma;
                                        Min=y;
                                        }
                                }
                        aux=10000;
                        for (z=0;z<maq;z++)
                                {
                                if (z!=Min)
                                        {
                                        Soma=Ds[z]+Ex[z][x];
                                        if (Soma<aux) // 2° menor data de conclusão
                                                {
                                                aux=Soma;
                                                SMin=z;
                                                }
                                        }
                                }
                        S[x]=(Ex[SMin][x]+Ds[SMin])-(Ex[Min][x]+Ds[Min]); // Índice S
                        if (S[x]>aux1) // Tarefas que vão ser alocadas
                                {
                                J=x;
```

```
                                aux1=S[x];
                                }
                        }
                }
        T[J]=0; // Tarefa vai ser alocada
        aux=10000;
        for (y=0;y<maq;y++)
                {
                Soma=Ds[y]+Ex[y][J];
                if (Soma<aux)
                        {
                        aux=Soma;
                        Min=y;
                        }
                }
        Ds[Min]=Ds[Min]+Ex[Min][J];
        for (y=0;y<maq;y++)
                {
                Ex[y][J]=0; // Alocação
                }
        Ex[Min][J]=1;
        F--;
        }
stop=clock(); // Para clock
dur=(double)stop-start;
printf ("\nAlocacao");
for (x=0;x<tar;x++)
        {
        printf ("\n");
        for (y=0;y<maq;y++)
                {
                printf ("%f ", Ex[y][x]); // Imprime alocação
                }
        }
printf ("\n\nDisponibilidades\n");
for (y=0;y<maq;y++)
        {
        printf ("%f ", Ds[y]); // Imprime disponibilidades
        }
```

```
aux=-1;
for (y=0;y<maq;y++)
		{
		temp=Ds[y];
		if (temp>aux)
			{
			aux=temp; // Encontra makespan
			}
		}
printf ("\n\nMakespan\n");
printf ("%f", aux); // Imprime makespan
printf ("\n\nDuracao\n");
printf("%f",(dur/CLOCKS_PER_SEC)*1000); // Imprime tempo
af=fopen("4Hipotese.txt", "w");
fprintf (af, "\nAlocacao");
for (x=0;x<tar;x++)
	{
	fprintf (af, "\n");
	for (y=0;y<maq;y++)
		{
		fprintf (af, "%f ", Ex[y][x]); // Imprime alocação
		}
	}
fprintf (af, "\n\nDisponibilidades\n");
for (y=0;y<maq;y++)
	{
	fprintf (af, "%f ", Ds[y]); // Imprime disponibilidades
	}
aux=-1;
for (y=0;y<maq;y++)
		{
		temp=Ds[y];
		if (temp>aux)
			{
			aux=temp; // Encontra makespan
			}
		}
fprintf (af, "\n\nMakespan\n");
fprintf (af, "%f", aux); // Imprime makespan
```

```
fprintf (af, "\n\nDuracao\n"); // Imprime tempo

fprintf(af, "%f",(dur/CLOCKS_PER_SEC)*1000);

fclose(af);

getchar();

getchar();

return 0;

}
```

Anexo II

1º Grupo de problemas.

Problem	Machine	Job								
Size		1	2	3	4	5	6	7	8	9
2X5	1	24	4	5	6	12				
	2	18	2	19	1	3				
2X6	1	23	10	22	9	1	22			
	2	9	9	1	9	16	15			
2X7	1	5	4	14	4	22	8	21		
	2	9	10	21	5	19	6	11		
2X8	1	9	13	9	5	23	16	23	19	
	2	19	23	3	14	8	14	9	13	
2X9	1	19	18	16	18	12	20	2	17	22
	2	2	1	12	10	21	3	14	19	10
3X5	1	21	19	9	16	3				
	2	6	24	19	2	19				
	3	18	4	9	21	4				
3X6	1	3	23	1	24	17	11			
	2	2	7	14	15	9	7			
	3	4	2	14	6	22	8			
3X7	1	17	12	20	2	4	21	9		
	2	18	13	9	14	2	18	2		
	3	7	11	14	8	18	20	11		
3X8	1	23	12	11	4	23	7	18	10	
	2	13	19	21	20	4	14	12	2	
	3	24	12	1	15	6	12	12	22	
3X9	1	15	5	11	19	22	12	8	13	17
	2	6	24	6	22	14	7	17	18	1
	3	9	2	16	2	20	10	2	20	5
4X5	1	24	16	10	22	14				
	2	11	12	10	21	4				
	3	22	17	17	2	10				
	4	18	5	1	23	3				
4X6	1	5	9	4	2	10	17			
	2	17	7	1	10	23	12			
	3	16	1	23	4	1	2			
	4	18	7	7	23	15	12			
4X7	1	11	8	1	14	7	17	8		
	2	24	13	23	5	20	21	13		
	3	20	24	20	19	23	9	12		
	4	5	10	21	16	12	21	20		
4X8	1	22	9	16	15	12	22	5	16	
	2	2	2	16	7	19	14	7	15	
	3	8	13	18	14	24	5	24	9	
	4	4	4	19	3	13	5	6	21	
4X9	1	8	11	22	7	1	8	18	19	1
	2	8	19	10	22	20	20	7	11	18
	3	7	11	17	5	11	7	7	23	22
	4	9	22	21	7	2	8	6	6	22
5X5	1	10	4	23	7	10				
	2	20	1	24	9	5				
	3	8	2	15	2	2				
	4	19	16	18	1	11				
	5	11	2	5	20	1				
5X6	1	22	8	4	7	5	4			
	2	7	2	9	7	5	14			
	3	19	2	5	17	5	14			
	4	12	3	13	5	12	9			
	5	3	13	3	23	5	4			
5X7	1	17	1	23	5	6	10	5		
	2	15	10	8	7	19	2	5		
	3	1	10	20	3	24	22	23		
	4	18	22	7	7	3	12	8		
	5	5	13	3	3	14	6	12		
5X8	1	13	6	23	18	11	17	5	12	
	2	18	19	10	16	23	2	23	24	
	3	16	5	2	7	22	9	4	8	
	4	20	10	23	24	18	18	23	2	
	5	4	24	13	12	13	10	23	10	
5X9	1	15	24	11	16	14	6	5	19	7
	2	2	21	16	17	2	15	7	10	13
	3	19	16	22	12	3	6	10	5	21
	4	7	20	23	15	17	19	12	19	2
	5	5	15	3	22	6	9	17	2	15

2º Grupo de problemas.

Problem	Machine	Job								
Size		1	2	3	4	5	6	7	8	9
2X5	1	3	9	5	7	14				
	2	24	23	15	11	5				
2X6	1	5	11	20	11	10	5			
	2	1	4	5	6	4	23			
2X7	1	20	16	5	1	23	13	6		
	2	5	1	6	4	18	24	15		
2X8	1	24	13	21	1	12	19	3	10	
	2	21	9	7	11	20	22	23	5	
2X9	1	17	18	6	4	5	10	19	5	24
	2	15	24	2	16	20	18	1	7	15
3X5	1	8	18	8	2	16				
	2	3	5	11	24	13				
	3	7	19	11	23	17				
3X6	1	21	14	11	14	7	12			
	2	8	12	6	10	8	6			
	3	16	12	19	13	10	17			
3X7	1	1	5	1	5	6	14	14		
	2	24	5	4	11	7	13	15		
	3	2	23	10	16	3	11	21		
3X8	1	15	17	15	1	2	2	5	7	
	2	19	8	1	15	15	15	5	8	
	3	4	6	15	5	7	17	20	8	
3X9	1	3	21	20	1	23	17	15	2	22
	2	5	12	2	8	16	22	6	24	16
	3	11	8	21	23	3	20	19	3	23
4X5	1	21	24	21	10	9				
	2	24	7	22	20	6				
	3	5	5	12	14	8				
	4	22	22	7	15	1				
4X6	1	17	4	1	7	3	11			
	2	19	23	12	24	17	4			
	3	1	18	7	23	3	18			
	4	5	18	6	22	3	18			
4X7	1	5	11	3	5	18	7	23		
	2	2	12	21	19	5	15	4		
	3	8	8	1	23	21	12	11		
	4	6	17	21	5	6	5	7		
4X8	1	22	4	15	15	22	15	6	19	
	2	2	1	3	16	5	15	12	20	
	3	2	5	16	17	5	24	1	5	
	4	16	13	24	9	5	2	1	6	
4X9	1	10	3	6	1	5	16	3	9	14
	2	16	5	5	11	24	18	11	7	1
	3	15	7	13	15	17	10	8	17	1
	4	21	22	20	19	12	11	19	1	8
5X5	1	14	13	4	11	9				
	2	22	6	7	20	20				
	3	23	4	19	13	2				
	4	18	5	23	21	21				
	5	17	8	16	18	15				
5X6	1	5	6	6	11	15	8			
	2	15	14	21	5	8	13			
	3	17	17	7	22	10	8			
	4	1	20	8	22	17	5			
	5	10	1	13	4	7	6			
5X7	1	9	10	17	3	20	2	19		
	2	18	23	2	22	9	5	1		
	3	20	11	15	4	19	20	24		
	4	24	2	2	17	7	22	12		
	5	14	20	2	15	24	11	22		
5X8	1	20	7	4	21	12	11	6	4	
	2	5	21	12	7	10	15	19	10	
	3	19	18	21	20	15	5	15	21	
	4	16	18	6	19	16	22	22	1	
	5	16	13	8	24	12	11	7	6	
5X9	1	13	1	14	4	11	12	21	20	14
	2	2	4	14	8	24	9	22	23	21
	3	4	8	5	8	22	20	1	6	5
	4	2	16	4	6	15	9	13	24	11
	5	13	15	14	13	17	22	13	9	5