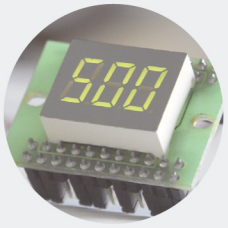
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






HEURÍSTICAS DE PESQUISA LOCAL PARA PROBLEMAS DE MÁQUINA ÚNICA

JOSÉ ALEJANDRO DE SOUSA MARQUES
Novembro de 2015

POLITÉCNICO
DO PORTO

HEURÍSTICAS DE PESQUISA LOCAL PARA PROBLEMAS DE MÁQUINA ÚNICA

José Alejandro de Sousa Marques

Departamento de Engenharia Electrotécnica

Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização em Sistemas e Planeamento Industrial

2015

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de Tese/Dissertação do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: José Alejandro de Sousa Marques, Nº 1080487, 1080487@isep.ipp.pt
Orientação científica: Ana Maria Dias Madureira Pereira, amd@isep.ipp.pt

Departamento de Engenharia Electrotécnica
Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Sistemas e Planeamento Industrial

2015

Agradecimentos

Queria prestar os meus agradecimentos ao ISEP pela formação prestada, e à Engenheira Ana Maria Dias Madureira Pereira pelo apoio e orientação científica prestados ao longo da elaboração deste projecto.

Especialmente aos meus pais, irmãos e avós pelo apoio incondicional dado ao longo deste percurso, e finalmente aos meus colegas de curso pelo apoio mostrado ao longo do meu percurso académico.

Resumo

O escalonamento é uma das decisões mais importantes no funcionamento de uma linha de produção. No âmbito desta dissertação foi realizada uma descrição do problema do escalonamento, identificando alguns métodos para a optimização dos problemas de escalonamento. Foi realizado um estudo ao caso do problema de máquina única através do teste de várias instâncias com o objectivo de minimizar o atraso pesado, aplicando uma Meta-Heurística baseada na Pesquisa Local e dois algoritmos baseados no SB.

Os resultados obtidos reflectem que os algoritmos baseados no SB apresentaram resultados mais próximos do óptimo, em relação ao algoritmo baseado na PL. Os resultados obtidos permitem sustentar a hipótese de não existirem algoritmos específicos para os problemas de escalonamento. A melhor forma de encontrar uma solução de boa qualidade em tempo útil é experimentar diferentes algoritmos e comparar o desempenho das soluções obtidas.

Palavras-Chave

Escalonamento, Máquina Única, soma pesada dos atrasos pesados, Pesquisa Local, Meta-Heurística

Abstract

Scheduling is one of the most important decisions to be made in the operation of a production line. In this dissertation the scheduling problem is explored, identifying optimization methods used to solve scheduling problems. It was conducted a study to the single machine problem, in which minimizing the weighted tardiness was the objective. It was tested by solving different instances of the problem. The results were obtained applying a Local-Search Meta-Heuristic and two algorithms based on the Shifting Bottleneck procedure.

The results obtained with the SB based algorithms present solutions closer to the optimum, when compared with the results obtained with the LS Meta-Heuristic. The results obtained support the hypothesis that there are no specific algorithms for scheduling. The best way to achieve a quality result in a time limited situation is to implement different algorithms and evaluate the performance of each solution.

Keywords

Scheduling, Single-Machine, Weighted tardiness, Local Search, Meta-Heuristic

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

EDD - Earliest due date

LPT - Longest Processing Time

MH - Meta-Heurística

PL - Pesquisa Local

SB - Shifting Bottelneck

WT - Weighted tardiness

SPT - Shortest processing time

1. INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

Este projecto surgiu da necessidade de realizar um trabalho no âmbito do Mestrado em Sistemas e Planeamento Industrial, em que o tema do escalonamento assume uma importância relevante devido aos níveis de complexidade e importância que pode atingir em ambientes industriais. Através da realização deste trabalho de mestrado será possível explorar o tema do escalonamento em ambientes de máquina única (*single machine*), assim como a aplicação de metodologias baseadas em PL para a resolução dos problemas de escalonamento.

Também com a realização deste trabalho será possível aprofundar os conhecimentos ao nível de programação em Linguagem C, na medida que será desenvolvido um protótipo que permita a resolução dos problemas de escalonamento em ambientes de máquina única, recorrendo a técnicas baseadas em Pesquisa Local.

1.2. OBJECTIVOS

O objectivo principal deste projecto é a análise do desempenho de técnicas baseadas em Pesquisa Local na resolução de problemas de escalonamento em ambientes de máquina única. Dada a complexidade inerente a este objectivo, sentiu-se a necessidade de o subdividir em múltiplas tarefas de realização mais simples, tais como:

- Pesquisa bibliográfica referente ao escalonamento da produção;
- Pesquisa bibliográfica referente às técnicas baseadas em Pesquisa Local (Meta-Heurísticas);
- Desenvolvimento do protótipo da Ferramenta de optimização baseada na Pesquisa Local;
- Análise do desempenho dos métodos aplicados.

1.3. ORGANIZAÇÃO DO RELATÓRIO

No Capítulo 1 é estabelecido o enquadramento do trabalho, também são apresentados os objectivos pretendidos com a realização do mesmo. No capítulo 2 é apresentando o problema do escalonamento, são introduzidos os sistemas de produção, a respectiva classificação, é introduzido o conceito de optimização combinatória e finalmente é descrito o problema de máquina única. No capítulo 3 são apresentados os sistemas de escalonamento, os diferentes tipos e são apresentados alguns exemplos de sistemas de escalonamento disponíveis. O capítulo 4 é dedicado aos métodos de optimização, a sua classificação e são descritos alguns dos métodos mais utilizados. É dado um foco especial às Meta-Heurísticas baseadas em Pesquisa Local. No capítulo 5 é apresentada a ferramenta de apoio desenvolvida no âmbito deste trabalho. No capítulo 6 são apresentados os resultados da aplicação da meta-Heurística desenvolvida, assim como uma comparação entre os métodos aplicados recorrendo ao Lekin. O capítulo 7 apresenta as principais conclusões e trabalho futuro.

2. PLANEAMENTO E ESCALONAMENTO DA PRODUÇÃO

2.1. INTRODUÇÃO

Actualmente as organizações têm de estar permanentemente a tomar decisões, seja a nível produtivo ou a nível de recursos humanos. Quando o sector produtivo da empresa planeia o que produzir na semana corrente, está a tomar uma decisão que pode comprometer o futuro da empresa, caso esta seja mal planeada. A falta de um planeamento correcto em relação às actividades diárias das organizações compromete os níveis de competitividade da mesma, o que se pode traduzir numa perda da quota de mercado. Por isso, o que produzir, quando e em que quantidades é uma forma de escalonar e que está ligado ao processo da tomada de decisão. O processo de escalonamento, dentro de uma organização, tem como principal objectivo a alocação dos recursos necessários para o seu correcto funcionamento. Para realizar o escalonamento dos recursos disponíveis na organização recorre-se a algoritmos desenvolvidos para o efeito e cuja complexidade pode depender de muitos factores, como por exemplo: disponibilidade dos recursos, duração das tarefas, os prazos de entrega, as precedências entre tarefas, etc., o que pode tornar o processo de escalonamento bastante

complexo. Basicamente o problema do escalonamento consiste na alocação dos recursos para determinada tarefa, num período de tempo, e o propósito é o de optimizar uma ou várias função objectivo. As tarefas que vão ser alvo da alocação de recursos dependem da natureza da organização. Pode-se tratar de alocar máquinas, em ambientes workshop, pistas de aterragem num aeroporto, equipas de trabalho num local de construção, etc (Baker, 2009), (Pinedo, 2009). As tarefas poderão ser operações ou tarefas que a máquina deve assumir, aterragens e descolagens num Aeroporto, fases da construção de uma infra-estrutura, entre outras. Os recursos a escalonar podem ter um nível de prioridade associado, uma data de início prevista ou uma data de entrega a cumprir, ou ambos se for o caso. A função-objectivo poderá ser de maximização (em caso de lucro a obter com o processo de escalonamento, por exemplo) ou de minimização (tempo de entrega de um produto associado a uma máquina). Como se pode observar são muitas as variáveis que condicionam uma correcta política de escalonamento, mas o conceito assume uma grande importância na indústria, nomeadamente nos processos produtivos, onde um escalonamento correcto das actividades que uma linha de produção deve executar dita o sucesso ou não da organização no seu dia-a-dia. Para que um conjunto de tarefas seja escalonada da forma mais fiável possível é necessário conhecer o tipo e a disponibilidade de recursos, isto permite também estabelecer de uma forma mais efectiva os limites do sistema, o que permite uma escolha do método apropriado às características do sistema. No entanto, em alguns tipos de problemas de escalonamento, nem sempre é possível determinar a duração de algumas das tarefas que devem ser executadas, pelo que deve-se tentar suprimir essa incerteza, mediante o desenvolvimento de algoritmos que se adaptem às características particulares do sistema. Existem modelos formais que permitem visualizar um problema de escalonamento e de seguida obter a sua resposta. O modelo mais simples e um dos mais utilizados é o Diagrama de *Gantt*. Basicamente permite visualizar a distribuição das tarefas ao longo do tempo, permitindo rearranjar a distribuição e analisar as possíveis consequências. Para a sua elaboração parte-se do pressuposto que o tempo de duração das tarefas é conhecido. As tarefas aparecem no eixo vertical do diagrama e o tempo em que estas devem ser executadas no eixo horizontal (Baker,2009).

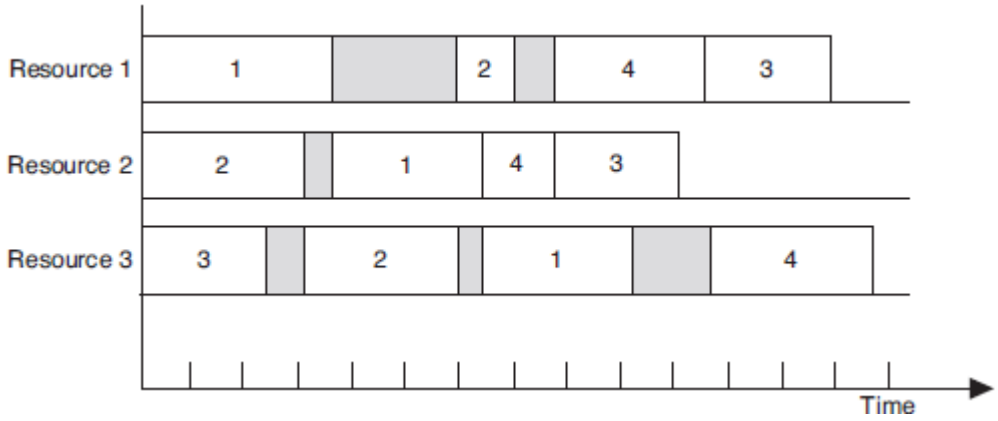


Figura 1 - Diagrama de Gantt (Baker, 2009)

Existem modelos matemáticos baseados em heurísticas que afectam a quantidade de recursos disponíveis no processo produtivo (principalmente) por forma a satisfazer o objectivo da função-objectivo no problema de escalonamento.

Os problemas de escalonamento são constituídos basicamente por uma série de restrições e uma função-objectivo que deriva dos objectivos traçados para o problema e dos recursos disponíveis para a atingir, que tal como foi mencionado previamente o objectivo pode ser a maximização ou minimização. Será da responsabilidade do algoritmo que corre por trás desta função encontrar a solução óptima do mesmo. No caso de sistemas de máquina única, o objectivo do escalonamento poderá ser o de reduzir o tempo de fluxo das actividades, reduzir o tempo de atraso de entrega das tarefas, reduzir a penalização monetária inerente aos atrasos, entre outros.

Para atingir esses objectivos traçados é preciso respeitar algumas restrições que são colocadas, e que vão depender quase exclusivamente da natureza do problema, que no âmbito de problemas de escalonamento em ambientes de produção, normalmente estão relacionadas com a quantidade de recursos que devem ser alocados assim como as datas nas quais as tarefas têm de ser executadas. A solução encontrada pelo algoritmo terá de conciliar estas duas restrições, podendo-se dizer que os problemas de escalonamento requerem que duas decisões sejam tomadas: a alocação dos recursos e a sequência pela qual as tarefas devem ser executadas.

O escalonamento tem sido um dos campos, a nível de produção, onde tem ocorrido grandes avanços ao longo da história. A capacidade dos sistemas computacionais tem vindo a aumentar no tempo, o que implica o aumento da possibilidade de implementar técnicas de

escalonamento que permitem obter resultados bastantes atractivos para as organizações. No entanto nem sempre é possível encontrar soluções óptimas para alguns dos problemas de escalonamento, devido ao esforço computacional que tal envolve, é preciso por vezes um compromisso no qual é possível obter uma solução boa e de acordo com os objectivos da empresa sem que envolva um grande esforço computacional.

2.2. SISTEMAS DE PRODUÇÃO

Um sistema de produção pode ser definido como um conjunto de elementos interrelacionados cujo objectivo passa por converter um conjunto de entradas nas saídas pretendidas mediante um processo de transformação. Os elementos podem ser máquinas, operadores ou ferramentas. As entradas podem ser a matéria-prima necessária, ou um produto acabado de outro sistema. As saídas podem ser os produtos acabados e informação relacionadas com eles.

O processo de transformação corresponde a uma série de operações onde correspondendo a um passo do processo global do fabrico de um produto. A gestão de um sistema de produção consiste basicamente em gerir os fluxos de materiais e de informação, controlando desta forma o processo de transformação da matéria-prima e o desempenho do sistema. Para que esta gestão seja realizada da melhor forma é necessário realizar uma compilação de toda a informação referente ao sistema para facilitar a tomada de decisões correctas para que os objectivos definidos sejam atingidos da melhor forma. Na Figura 2 é possível observar um exemplo de um sistema de produção. O planeamento, como se pode verificar, é um passo importante na elaboração de qualquer produto. Permite saber com antecedência necessidades da empresa no que se refere a materiais, à capacidade de produção, custos, entre outras.

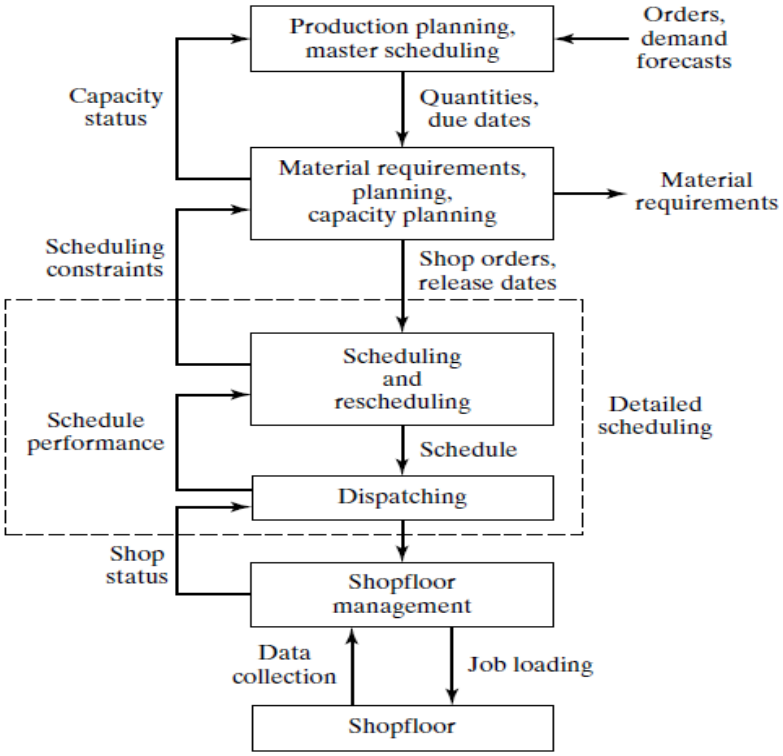


Figura 2 - Sistema de produção (Pinedo, 2009)

2.2.1. CLASSIFICAÇÃO DOS SISTEMAS DE PRODUÇÃO

A literatura refere várias formas de classificação dos sistemas de produção. Esta pode ser realizada considerando alguns parâmetros ou critérios. Por exemplo:

- Implantação dos recursos produtivos;
- Fluxos de materiais;
- Relação com o cliente;
- Quantidades fabricadas dum mesmo produto;
- A tipologia da estrutura dos produtos;
- A variabilidade dos produtos produzidos;
- Gama operatória;
- Natureza dos produtos;
- Condições do mercado;
- Organização;
- E produção no espaço.

Tabela 1 - Classificação dos sistemas de produção (Cavaco, 2008)

Implantação	Por projecto
	Por processo
	Células de fabrico
	Sistemas Flexíveis de Fabrico
	Por produto
Fluxo dos Materiais	Intermitente
	Contínuo
	Misto
Relação com o cliente	Engenharia por encomenda
	Fabrico por encomenda
	Montagem por encomenda
	Fabrico para stock
Quantidades produzidas do mesmo produto	Unitária
	Pequenas séries
	Grandes séries
	Séries constantes
Tipologia	Convergente
	Divergente
	Estrutura em T
Variabilidade	Produto Único
	Semelhantes
	Diferenciados
Gama Operatória	Única
	Semelhante
	Diferenciados
Natureza dos produtos	Discreta
	De processo
Caracterização da procura	Procura imprevisível
	Procura Variável
	Procura estável
Organização	Rígida
	Flexível
Produção no Espaço	Distribuída
	Concentrada

Na Tabela 1 é possível observar alguns dos critérios utilizados para classificar um sistema de produção. Tomando como exemplo o critério pela **relação com o cliente (Cavaco, 2008)**:

- Engenharia por encomenda: O cliente fornece as especificações, cabendo à empresa executar as directrizes fornecidas pelo cliente;
- Fabrico por encomenda: A empresa recebe a encomenda do cliente, e depois é da sua responsabilidade executar e entregar a mesma;
- Montagem por encomenda: No seguimento de um catálogo constituído por uma variedade de modelos básicos, a montagem dos produtos só é executada quando chegar uma encomenda do cliente;
- Fabrico para Stock: A empresa produz e armazena os produtos, e fornece aos clientes quando é necessário;

No processo de escalonamento, podemos identificar três conceitos chaves que de certa forma concentram os esforços em grande parte da tomada de decisão no escalonamento:

- Maximizar a produtividade (*throughput):* Pretende-se maximizar o número de tarefas executadas por unidade de tempo;
- Reduzir o tempo requerido para completar uma determinada tarefa *(turnaround time);*
- O nível de conformidade com o qual uma tarefa é realizada dentro de um limite de tempo (*Timeliness).*

Na Figura 2 podemos observar um esquema de um sistema de produção no qual as ordens de fabrico recebidas são "traduzidas" em tarefas com uma data de entrega associada. Estas tarefas são processadas num centro de trabalho (*work center)* numa determinada sequência. No entanto, a execução pode ser adiada ou mesmo existir um "engarrafamento" do sistema se as máquinas estiverem ocupadas ou entrarem no sistema tarefas com prioridade superior. Também podem ocorrer eventos que não estão previstos, como a avaria de uma máquina, o que ocasiona o atraso do sistema produtivo.

Por isso, em ambientes de produção reais torna-se necessário o desenvolvimento de um escalonamento rigoroso e que permita ultrapassar os imprevistos que possam surgir e por sua vez permita manter, e talvez, aumentar a eficiência e manter o controlo das operações que devem ser executadas. No entanto, a oficina de produção não é o único com impacto no processo de escalonamento, que também é afectado pelo departamento encarregue de

efectuar o planeamento a médio e longo prazo dentro da organização. Cujo objectivo visa optimizar o processo produtivo com todos os benefícios que isto traz para a organização.

2.3. PLANEAMENTO DA PRODUÇÃO

Tradicionalmente os problemas de escalonamento têm sido abordados como problemas de optimização, que estão sujeitos a restrições e cujos elementos chaves são as máquinas e as tarefas (Baker, 2009). Em (Madureira, 2003) é referido que os problemas de escalonamento estão divididos, principalmente, em 4 fases:

- **Formulação:** É identificado o tipo de problema e determinado o critério ou critérios que encaminharão o processo da tomada de decisão;
- **Análise:** Consiste na realização de uma análise detalhada aos elementos do problema assim como a forma em que estes se interrelacionam;
- **Síntese:** São construídas soluções alternativas admissíveis dentro do contexto do problema em questão;
- **Avaliação:** É a fase na qual é realizada uma análise comparativa das soluções encontradas e é seleccionada aquela que representa a solução ideal para o problema, considerando os critérios estabelecidos anteriormente e que conduziram à mesma.

Tal como foi dito inicialmente, os problemas de escalonamento são tradicionalmente abordados como problemas de optimização sujeitos a restrições, que visam a alocação de recursos no tempo e o sequenciamento pela qual as tarefas devem ser executadas e neste contexto são desenvolvidos modelos matemáticos recorrendo a linguagens de programação cujo objectivo é encontrar a solução óptima, dentro de um leque de soluções admissíveis, para os problemas de escalonamento. A escolha do modelo que deve ser implementado baseia-se na topologia, natureza e função objectivo do problema.

Os problemas reais de escalonamento, dependendo da sua dimensão, são de resolução muito complexa. Em termos de complexidade são conhecidos como NP-completos (*"NP-complete")* e não tem sido possível ao longo da história encontrar um algoritmo para a resolução deste tipo de problemas e a convicção é de que não há. Grande parte dos problemas de escalonamento são incluídos na categoria de NP-difíceis ("*NP-Hard)* nos quais o tempo para a determinação de uma solução óptima aumenta exponencialmente com a dimensão do problema, (Baker, 2009) (Madureira, 2003).

Na literatura surgem os termos **sequenciamento** (“*sequencing”)* e **escalonamento** (“*scheduling”).* Pelo que é necessário estabelecer o significado de ambos dentro do contexto. Sequenciamento refere-se à sequência ou ordem pela qual as operações devem ser executadas. O escalonamento refere-se à afectação dos recursos no tempo e ao sequenciamento das operações. Pelo que um problema de escalonamento pode ser dividido em duas partes, na primeira é planeada uma sequência ou estabelecida uma forma de escolher a próxima tarefa. Na segunda parte é planeado um tempo inicio para cada tarefa e por vezes definido um tempo limite para completar a mesma. Este tipo de pensamento está presente nos problemas do dia-a-dia, como por exemplo, na planificação de um almoço em grupo.

Em ambientes industriais, os problemas de escalonamento são basicamente constituídos por um grupo de tarefas e uma quantidade limitada de recursos a ser alocados. Pelo que visam a afectação no tempo de recursos, muitas das vezes limitados, sujeitos a restrições básicas, tais como:

- Em qualquer instante nenhuma máquina executa mais do que uma tarefa.
- Nenhuma tarefa é processada em simultâneo em mais do que uma máquina.

Os problemas de escalonamento foram definidos tendo como referência a resolução de problemas em ambientes industriais, no entanto a sua aplicação pode ser considerada também para a área de prestação de serviços (Pinedo, 2009). Dentro desta categoria podemos enquadrar os seguintes tipo de problemas de escalonamento:

- Escalonar sessões de radioterapia em centros Oncológicos;
- Escalonar os horários das enfermeiras num hospital;
- Reparação de automóveis numa oficina;
- Sistema de reservas num restaurante.

Os tipos de problema que podem surgir dependem sempre da situação onde se encontrem inseridos, no entanto é claro que surgem em ambientes industriais (uma linha de montagem) ou na prestação de serviços. Os recursos e as tarefas que devem ser executadas podem assumir diferentes formas, conforme o ambiente em que estão inseridos.

.

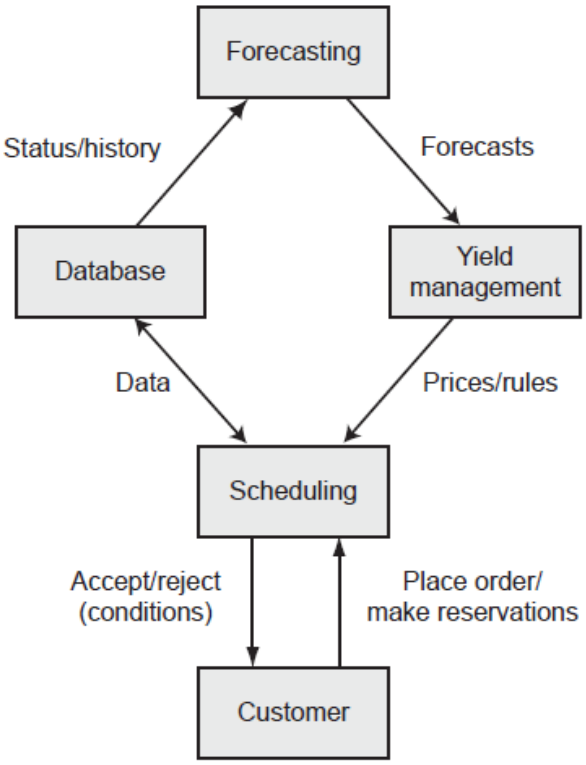


Figura 3 - Diagrama de fluxo de um sistema de serviço (Pinedo, 2009)

Na Figura 3, podemos observar o diagrama de fluxo de um sistema de serviços. Este tipo de problemas oferece diferentes tipos de desafios quando comparados com os problemas em ambientes industriais. São baseados em bases de dados extensivas que contêm toda a informação relevante aos recursos assim como as características dos clientes (actuais e futuros). Neste exemplo, podemos observar que o sistema de escalonamento tem de interagir com um módulo de gestão do rendimento assim como um módulo de previsão.

Em ambientes de produção industrial existem muitas situações passíveis de ocorrer e que podem comprometer o cumprimento do escalonamento definido. Avarias inesperadas, entrada no sistema de tarefas com precedências e por isto com uma prioridade maior sobre o resto, tempos de processamento mais demorados do inicialmente definido, entre outras.

Neste tipo de situações, é necessária uma interacção entre o departamento responsável pelo processo de escalonamento e o departamento responsável pelo planeamento da produção. As ferramentas utilizadas neste contexto são o MRP (*Material Requirements Planning)* ou Planeamento das necessidades de materiais e o ERP (*Enterprise Resource Planning)* ou Sistema integrado de gestão empresarial. Depois de realizado o escalonamento, é necessário garantir que os materiais se encontram disponíveis nas datas específicas.

Actualmente as empresas utilizam sistemas de planeamento elaborados, envolvendo uma rede de computadores e bases de dados que permitem uma constante monitorização do estado dos sistemas de produção. Os computadores em cada estação encontram-se ligados entre si através de um servidor, que pode ser utilizado para monitorizar e recolher informação das diferentes bases de dados ou para adicionar nova informação. É uma medida necessária que conjuga os diferentes departamentos de uma organização envolvidos na tomada de decisão tornando o processo de escalonamento, planeamento e produção mais eficientes. A Figura 4 mostra como todos estes componentes se encontram ligados entre si.

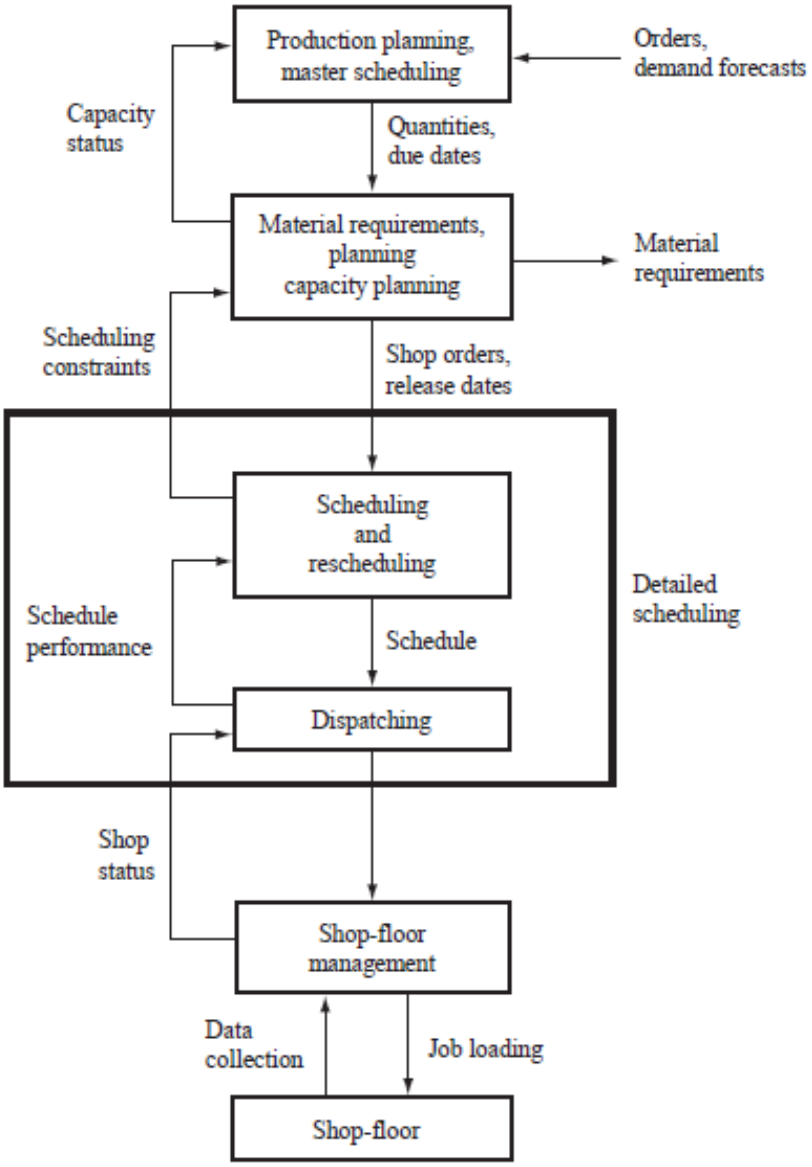


Figura 4 - Diagrama de fluxo de um sistema de produção (Pinedo, 2009)

2.4. ELEMENTOS DOS PROBLEMAS DE ESCALONAMENTO

Como referido anteriormente, os problemas de escalonamento tem sido abordados tradicionalmente como problemas de optimização. Os elementos básicos que constituem um problema de escalonamento são as máquinas e as tarefas (Madureira, 2003). Os problemas to tipo máquina única, abordado neste trabalho enquadram-se neste contexto, em que são escalonadas as tarefas que a máquina deve executar. Numa perspectiva geral, em relação aos problemas de escalonamento, podem ser identificados 3 conceitos.

- **Máquina:** Identificada como recurso em (Pinedo, 2009). Podem ser classificadas através das suas capacidades funcionais ou qualitativas e quantitativas.
- **Tarefa:** É composta por um conjunto de operações que devem ser executadas para a produção de um produto ou lote de produtos. É executada nas máquinas.
- **Operação:** É o processo de transformação de uma tarefa a ser processada numa máquina. Tem associado um tempo de processamento.

2.5. OBJECTIVOS E MODELOS

O problema de escalonamento mais elementar consiste num conjunto de tarefas uni-operação a ser executados numa máquina, sendo que os tempos de processamento e data de entrega são conhecidos previamente e independentes da sequência na qual são processados. (Madureira, 2003)

> A teoria do escalonamento preocupa-se principalmente com o desenvolvimento de modelos e algoritmos que permitam a resolução dos problemas. A perspectiva teórica é uma aproximação quantitativa, que permita descrever a estrutura do problema numa forma matemática. Esta aproximação quantitativa começa com a descrição dos recursos disponíveis e as tarefas e com a concretização dos objectivos traçados na tomada de decisão numa função objectivo explícita (Baker, 2009). Idealmente a função objectivo engloba todos os custos associados ao processo de escalonamento, no entanto na prática são difíceis de medir ou mesmo identificar.

São identificados três tipos de objectivos gerais que prevalecem nos problemas de escalonamento:

- O tempo requerido para completar uma tarefa;
- O cumprimento dos prazos de entrega;
- Uma produtividade eficiente. Maximizar a quantidade de trabalho finalizada durante um período fixo de tempo.

Os modelos de escalonamento são categorizados através da configuração dos recursos e a natureza das tarefas. Um modelo de escalonamento pode ser constituído apenas por uma máquina ou várias. No primeiro caso é provável que as tarefas sejam executadas apenas numa estação, no segundo caso é provável que as tarefas sejam executadas em diferentes máquinas. Também é referido que se o número de tarefas a executar é constante ao longo do tempo, estamos perante um sistema estático. Se a quantidade de tarefas é variável, isto é, novos trabalhos aparecem para serem executados, estamos perante um sistema dinâmico. Se estivermos perante um modelo, no qual se assume que as condições são conhecidas com certeza, estamos perante um modelo determinístico. Se pelo contrário, forem identificadas incertezas com distribuições estatísticas explícitas, estamos perante um modelo estocástico (Baker, 2009).

2.6. MODELOS TEÓRICOS *VERSUS* MODELOS REAIS

Os problemas de escalonamento que surgem nos sistemas de fabrico são muito complexos. A sua natureza é classificada como não-determinística, discreta, dinâmica e distribuída. Os modelos tratados na literatura enquadram-se na categoria de modelos teóricos. As diferenças que existem são diversas, nomeadamente (Madureira, 2003):

- Os problemas teóricos assumem que não existem tarefas em curso de fabrico (sistema vazio). É considerado que existem *n* tarefas para escalonar e que após a criação do plano de escalonamento para estas tarefas, o problema fica resolvido. Em sistemas de fabrico real estes pressupostos não se verificam, não só existem tarefas em curso, como chegam novas tarefas para ser introduzidas no sistema de forma contínua e muitas das vezes de forma aleatória. Como foi mencionado anteriormente, recorre-se a sistemas de previsões que permitem ultrapassar em certa medida as perturbações de natureza aleatória;

- Os problemas teóricos não consideram o problema do reescalonamento. O plano de escalonamento dos problemas teóricos é constituído tomando como base alguns pressupostos, na realidade podem surgir elementos aleatórios requerendo modificações mais ou menos significativas. O reescalonamento é feito tomando em consideração o cumprimento de algumas restrições;
- Nos ambientes de produção existem restrições que não são consideradas nos modelos teóricos:
    - Restrições a nível de precedências;
    - Restrições a nível de recursos e consequentemente execução das tarefas;
    - Restrições nas datas de lançamento e datas de entrega.
- As prioridades das tarefas assumidas na resolução de modelos teóricos são fixas. Na realidade a prioridade assumida no início pode variar ao longo do tempo, e uma tarefa com prioridade baixa pode assumir um papel de maior importância a qualquer instante.

Existem portanto muitas diferenças entre os problemas de escalonamento presentes no dia-a-dia e os problemas de escalonamento tratados em ambiente académico, os modelos teóricos. Nos modelos reais também pode ser incluído o erro humano, que pode derivar em consequências para o cumprimento do escalonamento programado.

2.7. COMPLEXIDADE COMPUTACIONAL

Os problemas de escalonamento enquadram-se no contexto dos problemas de optimização combinatória e são geralmente classificados de acordo com a teoria da complexidade computacional. O termo nasce nos primeiros anos da década do 70 e veio desempenhar um papel importante nas ciências da computação. Associado ao objectivo de classificar os problemas de optimização encontrados na prática como de fácil ou difícil resolução.

Na optimização combinatória é habitual falar-se em problemas de fácil e difícil resolução. Os primeiros são conhecidos como problemas de classe polinomial P, para os quais existe um algoritmo eficiente na sua resolução. Os segundos são classificados como NP entre os quais não existe nenhum algoritmo eficiente que os resolva em tempo útil, devido à sua complexidade exponencial. A abreviação NP surge do inglês "*nondeterministic polynomial time"*. Para problemas identificados como NP não existem algoritmos eficientes para a sua resolução, recorrendo-se portanto a métodos de aproximação que conseguem fornecer uma

resposta suficientemente boa, mas não óptima. Os problemas de optimização, na sua maioria, são incluídos na classe de problemas NP (Papadimitrou, 1998).

Em Papadimitrou (1998) é referido que um problema computacional é considerado satisfatoriamente resolvido quando o desempenho dos algoritmos conhecidos para a resolução deste problema não consomem muito tempo na procura da solução óptima. Um algoritmo é considerado eficiente na resolução de um problema se a sua complexidade crescer polinomialmente e não exponencialmente com a dimensão do problema.

Alguns exemplos de algoritmos que resolvem problemas de optimização à optimalidade em tempo polinomial são (Madureira, 2003):

- Métodos de transporte;
- Métodos de afectação;
- Método do caminho mais curto;
- Métodos de programação linear;

Nos problemas de optimização a complexidade pode ser descrita da seguinte forma (Madureira, 2003):

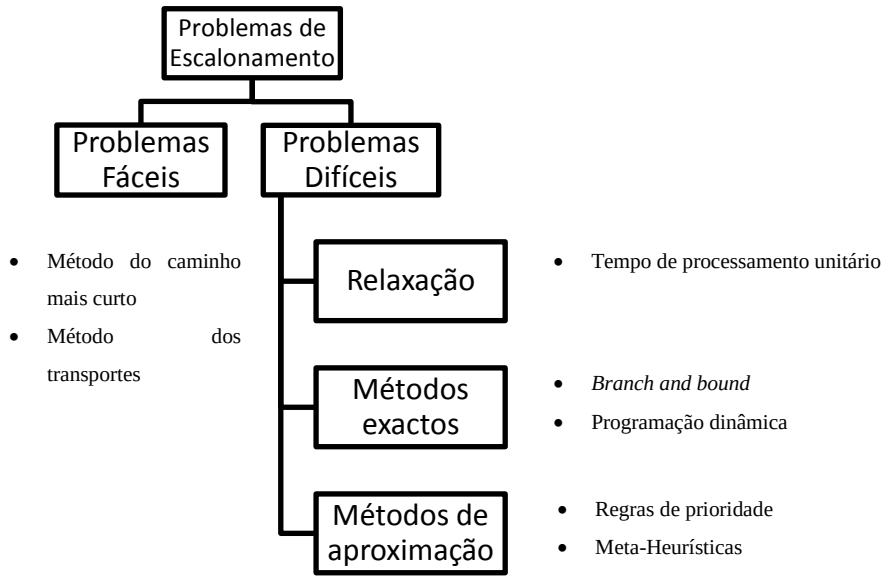


Figura 5 - Complexidade dos problemas de optimização (Madureira, 2003)

2.8. MEDIDAS DE DESEMPENHO

Os critérios de um plano de optimização estão normalmente associados às estratégias de negócio assumidas pela empresa. Estes critérios são definidos na função objectivo do problema e expressos na forma de minimização ou maximização, dependendo do tipo de variável. Normalmente variáveis associadas a penalizações, custos, tempos que demora uma sequência de tarefas a ser executada, atrasos médios, entre outros são variáveis a ser minimizadas. As variáveis associadas à maximização normalmente estão relacionadas com o lucro que a empresa obtém dos produtos.

Os planos de escalonamento são avaliados pela informação agregada resultante de todas as tarefas escalonadas. Este tipo de análise é denominado por medidas de desempenho e normalmente são funções que reflectem os tempos de conclusão do conjunto de tarefas escalonadas.

Em Madureira (2003) é referido que as medidas de desempenho podem ser agrupadas em três categorias. De referir:

- Baseadas nos tempos de desempenho;
- Baseadas nas datas de entrega;
- Baseadas nos custos de armazenamento.

Existem outras que podem ser analisadas como as relacionadas com o desempenho do sistema e os tempos de percurso.

As medidas de desempenho aplicadas dependem, primeiro, do tipo de sistema produtivo presente na empresa e, segundo, do modelo de gestão e objectivos financeiros da empresa. As medidas que para uns significam muito, para outros poderão não fazer grande sentido.

De seguida serão referidas algumas medidas de desempenho referidas em Madureira (2003), Baker (2009) e Pinedo (2009). Existe duas classes de medidas de desempenho, as regulares e as irregulares. As medidas analisadas de seguida são do tipo regulares, e são as que tem vindo a ser mais exploradas ao longo da história, visam a diminuição do tempo total de conclusão das tarefas.

No contexto dos sistemas produtivos e das medidas de desempenho apresentadas as variáveis são designadas da seguinte forma:

- p_j Tempo de processamento
- r_j Data de lançamento
- d_j Data de entrega
- C_j Tempo de conclusão

2.8.1. MEDIDAS DE DESEMPENHO RELACIONADAS COM O TEMPO DE PERCURSO

- **Tempo de percurso de uma tarefa**

O tempo de percurso F_j , em que *j* corresponde ao índice da tarefa, é o tempo transcorrido entre a data de lançamento no sistema e a data de conclusão da mesma

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

- **Tempo médio de percurso**

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

Onde *n* representa o número de tarefas

- **Tempo máximo de percurso**

Corresponde ao tempo de percurso da tarefa que permaneceu mais tempo no sistema até ser concluída.

$$F_{max} = max(F_j)\ \forall\ j \in \{1 \dots n\} \quad (3)$$

A taxa de produção de um sistema de fabrico está relacionada com os tempos de percurso. Quanto maior for a taxa de produção, menor será o tempo que estas permanecem no sistema (Madureira, 2003).

2.8.2. MEDIDAS DE UTILIZAÇÃO DE DESEMPENHO RELACIONADAS COM A UTILIZAÇÃO DO SISTEMA

- **Utilização de uma máquina**

Esta medida de desempenho é dada pela razão entre o tempo que a máquina é efectivamente utilizada e o tempo total em que essa máquina está disponível. Neste contexto C_{max} corresponde ao tempo total de conclusão das tarefas.

$$U_i = \frac{1}{C_{max}} \sum_{j=1}^{N} p_{ij} \times 100 \quad (4)$$

Onde p_{ij} representa o tempo de processamento de cada operação *j* na máquina *i*.

- **Utilização do sistema**

Esta medida representa a média da utilização de cada máquina do sistema:

$$U_i = \frac{1}{C_{max} \times M} \sum_{i=1}^{M} \sum_{j=1}^{N} p_{ij} \times 100 \quad (5)$$

M representa o número total de máquinas e p_{ij} é o tempo de processamento de cada operação *j* na máquina *i*

2.8.3. MEDIDAS DE DESEMPENHO RELACIONADAS COM AS DATAS DE ENTREGA

- **Atraso**

Mede a conformidade de conclussão das tarefas de uma determinada sequência com a sua respectiva data de entrega d_j

$$L_j = C_j - d_j \quad (6)$$

A terminologia encontrada na bibliografia para esta medida é "*lateness*". É de salientar que o atraso pode assumir valores negativos sempre que a tarefa é concluída mais cedo do que a respectiva data de entrega. Uma antecipação pode ser interpretado como um melhor serviço do que o requerido. Em determinadas situações existem penalizações para atrasos positivos, mas para atrasos negativos não existem benefícios directos. Na Figura 6, podemos observar como se comporta graficamente o atraso de uma tarefa face ao facto do tempo de conclusão ser superior à data de entrega.

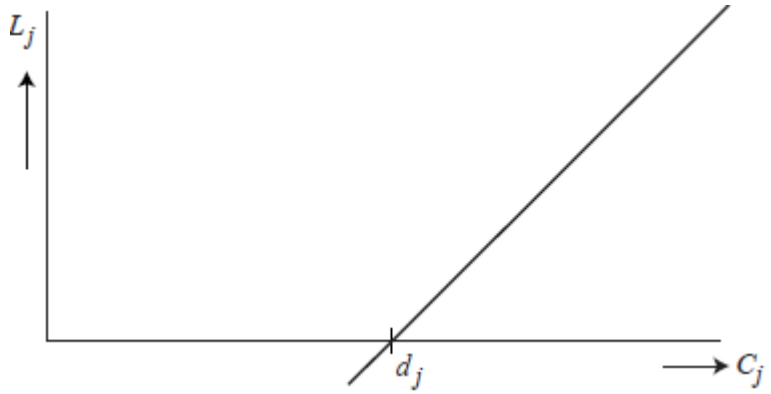


Figura 6 - Representação gráfica do atraso (Pinedo, 2009)

- **Atraso máximo**

Corresponde ao valor máximo assumido por L_j.

$$L_{max} = \max\{L_j\} \forall\, j\{1 \dots n\} \quad (7)$$

- **Atraso Positivo**

Representa um atraso efectivo na conclusão de processamento de uma tarefa. Na literatura é identificado por "*Tardiness*"

$$T_j = \max\{L_j, 0\} \quad (8)$$

Indica o número de unidades de tempo em que a data de entrega planeada é ultrapassada. O valor máximo de T_j designa-se por atraso máximo T_{max}.

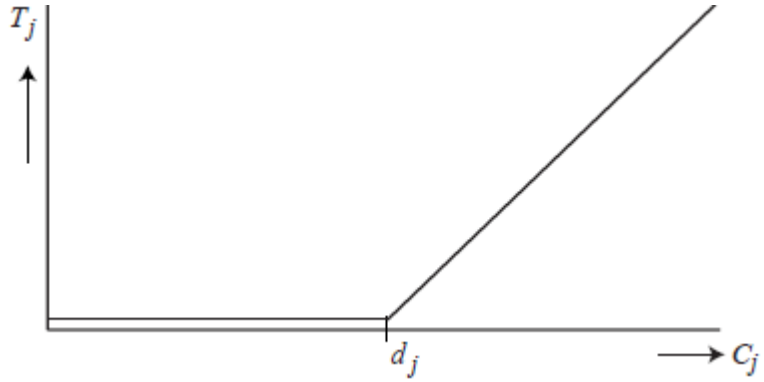


Figura 7 - Representação gráfica do atraso positivo (Pinedo, 2009)

- **Atraso negativo**

Representa uma antecipação na conclusão de processamento de uma tarefa. É identificado na literatura por "*Earliness"*.

$$E_j = \max\{0, -L_j\} \tag{9}$$

- **Atraso médio**

Representa a média dos atrasos das diferentes tarefas:

$$\bar{T} = \frac{1}{n}\sum_{j=1}^{n} T_j \tag{10}$$

- **Soma dos atrasos pesados**

Permite determinar a soma pesada dos atrasos, em que cada unidade de tempo em atraso de uma dada tarefa tem associada uma penalização $w_{j.}$ Na literatura é identificado por "*Weighted tardiness"*

$$WT = \sum_{j=1}^{n} w_j T_j, com\ T_j = \max\{L_j, 0\} \tag{11}$$

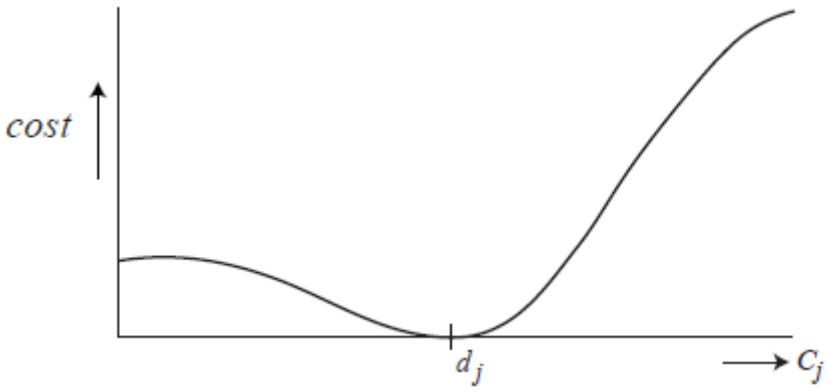


Figura 8 - Representação gráfica da soma dos atrasos pesados (Pinedo, 2009)

- **Número de tarefas em Atraso**

Através desta medida de desempenho, é possível determinar o número de tarefas em atraso em relação a respectiva data de entrega. Assume o valor 0 se o atraso L_j da tarefa j for negativo, e o valor 1 se L_j for positivo.

$$N_T = \sum_{j=1}^{n} f(L_j) \quad com\ f(L_j) = \begin{cases} 0\ se\ L_j \leq 0 \\ 1\ se\ L_j > 0 \end{cases} \tag{12}$$

- **Soma dos custos de posse e de atraso**

Esta medida permite determinar o custo total de posse e atraso. Pode-se utilizar em problemas nos quais se pretende encontrar uma sequência de taregas que minimize a soma dos custos de posse para as tarefas concluídas antes da data de entrega, assim como os custos de atraso para as tarefas concluídas após a referida data de entrega.

$$HT = \sum_{j=1}^{n} (h_j \times \max\{d_j - C_j, 0\} + w_j \times \max\{C_j - d_j, 0\}) \tag{13}$$

Onde C_j é o tempo de conclusão, h_j o custo de posse por unidade de tempo e w_j o custo de atraso por unidade de tempo para a tarefa *j*.

Por vezes é difícil estabelecer um critério de optimização satisfatório. Podem existir para a empresa vários objectivos que se pretendem atingir, conflituosos entre si e que não podem ser representados através de um critério único. Idealmente, deveriam ser considerados todos os objectivos relevantes para o processo de escalonamento, ainda que diferentes níveis de importância. Neste contexto é necessário atribuir prioridades ou graus de importância aos diferentes critérios ou definir critérios múltiplos.

Os critérios de optimização dependerão sempre da natureza do sistema produtivo e da estratégia de negócio assumida pela empresa, e estão relacionados com as medidas de desempenho referidas acima. Alguns destes critérios são:

- Minimização do tempo total de fabrico de um conjunto de tarefas. Também conhecido como "*makespan*";
- Cumprimento das datas de entrega;
- Maximização da utilização de recursos;
- Maximização da taxa de produção;

- Minimização do atraso total;
- Minimização do número de tarefas em atraso;
- Minimização do número de tarefas em curso de fabrico.

Neste contexto, a optimização de um dos critérios garante a optimização de outros. A minimização do tempo total de fabrico de um conjunto de tarefas em ambiente estático, equivale à maximização da utilização de recursos presentes no sistema.

2.9. CLASSIFICAÇÃO DOS PROBLEMAS DE ESCALONAMENTO

A complexidade do sistema de processamento permite avaliar o número de passos de processamento ou operações e máquinas associadas ao processo de escalonamento. Em Madureira (2003), Pinedo (2012) e Baker (2009) é possível identificar duas categorias na classificação dos problemas de escalonamento. Problemas com tarefas uni-operação e problemas com tarefas multi-operação.

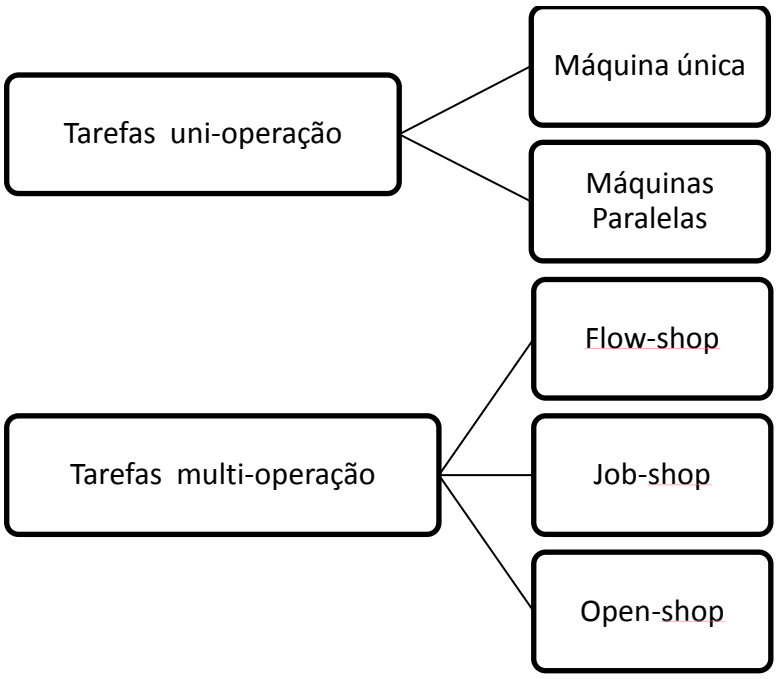


Figura 9 - Classificação do problema de escalonamento (Madureira, 2003)

- **Problemas uni-operação**

Tal como o seu nome o indica, neste tipo de problemas as tarefas a executar são constituídas por uma única operação, processada numa única máquina. Isto é aplicável a sistemas com máquina única ou sistemas com máquinas em paralelo. A diferença reside no facto de em sistemas de máquinas paralelas cada tarefa ser executada numa máquina, dentro de um conjunto de máquinas disponíveis.

Máquina única – O caso da máquina única é o problema de escalonamento mais elementar. O sequenciamento de *n* tarefas em máquina única determina completamente o plano de escalonamento. Embora na teoria pareça um modelo simples, é um modelo que pode ser utilizado em métodos de decomposição. Por exemplo, se num sistema com múltiplas máquinas existir um engarrafamento no sistema de produção, este determina o desempenho do sistema. Nesta situação, a abordagem mais correcta seria escalonar o sector onde ocorre o engarrafamento e de seguida escalonar o resto do sistema. Isto é possível de realizar ao reduzir o sistema a um problema de escalonamento de máquina única, permitindo resolver o problema do engarrafamento.

Máquinas paralelas – Um grupo de máquinas em paralelo é uma generalização do problema de máquina única. Muitos sistemas de produção são constituídos por várias fases ou *workcenters*, cada um com número de máquinas em paralelo associado. No contexto de um sistema com máquinas em paralelo, a determinação de um plano de escalonamento óptimo ou satisfatório torna-se, por vezes, um processo bastante complexo, dada a necessidade de tomar dois tipos de decisão: **afectação e sequenciamento.** Dentro do universo de máquinas paralelas é possível identificar 3 modelos possíveis (Pinedo, 2009):

- **Máquinas idênticas**: Neste modelo existem *n* máquinas idênticas em paralelo. A tarefa *j* pode ser processada, com a mesma velocidade, em qualquer uma das máquinas presentes no sistema. Assume-se que a máquina *i* processa a tarefa j com a mesma velocidade S_{ij}. Normalmente admite-se que o factor de velocidade de cada máquina, $S_{ij} = 1 \; \forall i, j$. A designação do termo velocidade corresponde à nomenclatura em inglês, *speed.*
- **Máquinas em paralelo com diferentes velocidades:** Existem *n* máquinas no sistema e cada uma apresenta uma velocidade diferente para o processamento de determinada tarefa, geralmente diferente de outras máquinas. Esta velocidade não depende da tarefa a processar, cada máquina *j* tem uma velocidade de execução

invariável, sendo normalmente o tempo de processamento da tarefa estabelecido por p_j/s_{ij} sendo p_j um valor constante para a tarefa j e S_{ij} o factor de velocidade associado à máquina i. Normalmente p_j refere-se ao tempo de processamento numa máquina para a qual se supõe S_{ij} igual à unidade. Este ambiente também é conhecido com máquinas uniformes. No caso da velocidade S_{ij} ser igual para todas as tarefas e o tempo de processamento $p_{ij} = p_j$ então estamos perante o primeiro caso.

- **Máquinas não relacionadas:** Este modelo é uma generalização dos primeiros. A velocidade de processamento de uma máquina depende da tarefa que está a ser executada, não estando relacionada com as outras máquinas. Neste caso o tempo de processamento de cada tarefa varia entre máquinas, mas de uma forma não relacionada entre elas.

- **Problemas com tarefa multi-operação**

Neste tipo de problemas, cada tarefa é constituída por um conjunto de operações processadas em máquinas diferentes. Para esta classificação de tarefas existem 3 configurações que o sistema pode assumir.

- **Flow-shop:** Existem n máquinas organizadas em série. Cada uma das tarefas tem de ser processada em todas as máquinas. Depois de acabar o se processamento numa das máquinas, fica numa fila a espera de ser processada pela máquina seguinte. O sequenciamento pelo qual as tarefas são executadas pode variar de máquina para máquina, isto deve-se ao facto de poder surgir um novo sequenciamento na máquina seguinte. Na Figura 10, é possível observar um exemplo de uma linha de produção configurada em flow-shop.

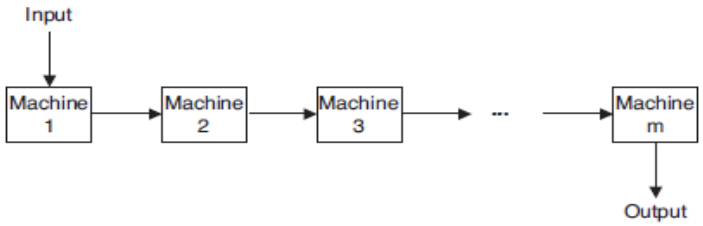


Figura 10 - Exemplo de um flow-shop (Pinedo, 2009)

- **Flow-shop flexível:** É um caso particular do flow-shop e uma generalidade do caso de máquinas paralelas. Em vez de máquinas em série existem *workcenters*, cada um deles com *n* máquinas em paralelo. Cada uma das tarefas tem de passar por cada um dos *workcenters* existentes no sistema. Na Figura 11, pode-se observar o esquema de um Flow-Shop flexível.

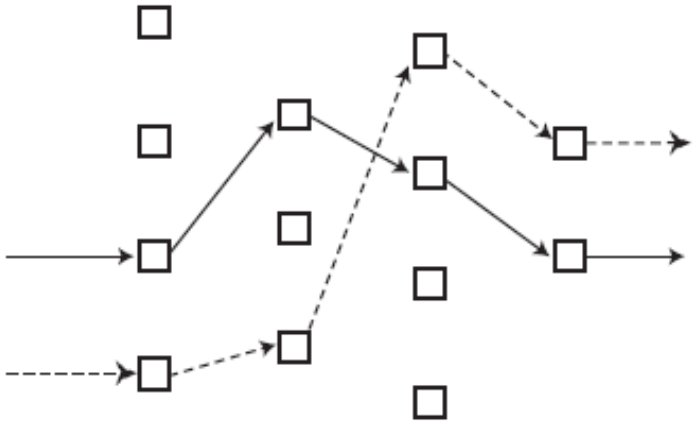


Figura 11 - Flow-shop flexível (Pinedo, 2009)

- **Job-Shop:** Consiste num ambiente de produção onde um conjunto de máquinas diferentes processa as operações das tarefas. Cada uma das tarefas que é executada tem uma sequência específica de processamento para percorrer as máquinas, ou seja cada tarefa é composta por um conjunto de operações, sequenciado, que se caracteriza pela máquina onde está a ser realizada e pelo tempo de processamento associado.

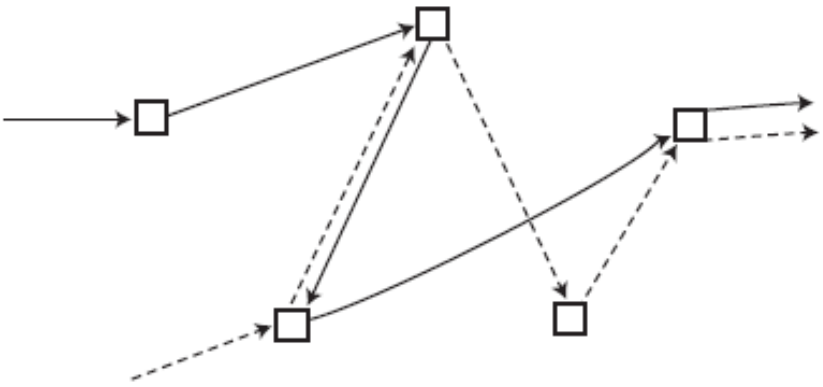


Figura 12 - Exemplo de um Job-Shop (Pinedo, 2009)

- **Job-shop flexível:** É um caso particular do job-shop e uma generalidade do caso de máquinas em paralelo. Em vez de existirem *n* máquinas em série existem *n* workcenters em paralelo uma disposição em série, mas máquinas encontram-se em paralelo. Cada uma das tarefas que é executada tem uma sequência específica de processamento para percorrer os *workcenter* e apenas uma máquina pode executar a tarefa em questão. De ressaltar que a tarefa pode ser executada por qualquer uma das máquinas presente no *workcenter*. Esta configuração é comum de se encontrar na indústria dos semicondutores.

- **Open-shop:** Este modelo é constituído por *n* tarefas a serem processadas por *m* máquinas diferentes. Não existe uma restrição em relação à ordem pela qual as tarefas devem ser executadas, sendo que o sequenciamento e escalonamento podem ser diferentes para cada uma das tarefas do sistema.

O ambiente de escalonamento no qual se encontra inserido o sistema permite saber se se trata de um problema **estático** ou **dinâmico** (Madureira, 2003), (Baker 2009). No primeiro caso é possível afirmar que se o número de tarefas disponíveis no início do processo de escalonamento não sofrer alterações durante o tempo, o problema é considerado **estático.** Se o número de tarefas definidas no início do problema variar com o tempo, isto é, aparecem novas tarefas, o problema é considerado **dinâmico.**

2.9.1. VARIABILIDADE DOS PARÂMETROS

Os parâmetros que fazem parte de um problema de escalonamento podem ser classificados de acordo com a sua variabilidade. Os modelos são classificados em Baker (2009), Pinedo (2009) e Pinedo (2012) em **Determinísticos** e **Estocásticos:**

- **Determinísticos:** Todos os parâmetros das tarefas são conhecidos à partida. Mesmo que existam dados de lançamento diferentes para cada tarefa, estes são conhecidos.
- **Não determinísticos:** Corresponde ao cenário real do problema de escalonamento. Nos ambientes reais há factores aleatórios que podem provocar alterações no estado do sistema. O plano de escalonamento precisa de ser reescalonado sempre que eventos dessa natureza aleatória se manifestarem no sistema. Nos problemas deste tipo existe incerteza em relação aos parâmetros, associados a uma função de distribuição probabilística.

Nos problemas de escalonamento estáticos, os parâmetros do problema são conhecidos no início do período de escalonamento, pelo que se considera que estes são implicitamente determinísticos. A classificação enquanto a variabilidade é justificável quando se está na presença de problemas de escalonamento dinâmicos. Os sistemas reais, presente nas empresas, são dinâmicos não-determinísticos, devido a que o seu estado é alterado pela possibilidade da ocorrência de eventos aleatórios que podem alterar o plano de escalonamento inicialmente definido (Madureira, 2003).

2.10. OPTIMIZAÇÃO COMBINATÓRIA

O conceito de Optimização Combinatória definiu-se como (Schrijver, 2002):

"*A Optimização Combinatória é a procura de um objecto óptimo numa colecção finita de objectos."*

Os problemas de optimização podem ser especificados por um conjunto de instâncias. A cada instância é associado um espaço de soluções possíveis. O conjunto de soluções, limitado pelas restrições do problema, cresce exponencialmente com a dimensão do problema, pelo que o conjunto de soluções admissíveis, embora finito, é de grande dimensão, não sendo uma opção analisar todos os objectos um a um para escolher aquele que representar a melhor solução.

De uma forma geral um problema de Optimização Combinatória pode ser representado da seguinte forma:

$$P: \max f(x)$$
$$Sujeito\ a\ x \in F \subseteq C$$

C representa o espaço de solução. Sendo que *F* representa o conjunto de soluções possíveis, sendo estas definidas pelas restrições do problema. Em Madureira (2003) é referido que os conjuntos *F* e *C* são discretos, e que podem ser definidos por um conjunto de variáveis de decisão. Estas variáveis podem assumir valores diferentes valores inteiros, dependendo do seu papel na formulação do problema.

Os problemas de Optimização Combinatória são enquadrados em duas categorias, com dificuldade polinomial P, para os quais existe um algoritmo que apresente uma solução óptima, em tempo útil e os de dificuldade NP, para os quais não existe um algoritmo que apresente uma solução óptima em tempo útil (a dificuldade cresce de forma exponencial

com a dimensão do problema e não de forma polinomial), pelo que são aplicados métodos de aproximação ou meta-heurísticas no qual a solução é satisfatória, mas não óptima. Na Figura 13, é possível observar alguns métodos de resolução aplicados aos problemas de Optimização Combinatória.

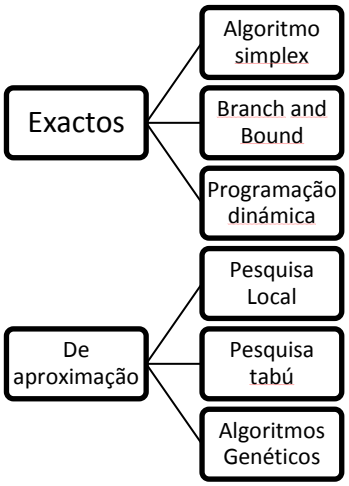


Figura 13 - Métodos de resolução para problemas de Optimização Combinatória

2.11. MÉTODOS DE RESOLUÇÃO DOS PROBLEMAS DE ESCALONAMENTO

Os problemas de escalonamento são, na maior parte dos casos, problemas complexos e de caracter combinatório. No entanto também existem instâncias dos problemas de escalonamento que são de fácil resolução, que podem ser formulados como um problema de programação linear e existem algoritmos que permitem a sua resolução em tempo útil e de forma eficiente. A sua dificuldade aumenta de forma polinomial e não de forma exponencial. A resolução de uma instância com centenas de tarefas por escalonar é possível em tempo útil num computador. Nos outros problemas de escalonamento, classificados como NP-difíceis, não podem ser formulados como um problema de programação linear e não existem algoritmos ou regras que possam encontrar uma solução óptima em tempo útil. Encontrar a solução óptima para problemas NP-difíceis não é uma tarefa fácil, pelo que tem de se considerar uma solução satisfatória para o problema em causa, gerada por um algoritmo em tempo útil e de forma eficiente (Madureira, 2003), (Pinedo, 2009).

Os métodos de resolução para problemas de optimização combinatória, categoria na qual se incluem os problemas de escalonamento, podem ser divididos em duas categorias abrangentes, os **métodos exactos** e os **métodos de aproximação**. Os primeiros consistem em encontrar uma solução óptima dentro do espaço de solução, através de uma pesquisa exaustiva. Nesta categoria podemos encontrar técnicas como o *"Branch and bound"* e a programação dinâmica. Os métodos de aproximação tem como objectivo encontrarem uma solução aceitável num período de tempo aceitável. Para a resolução de problemas de escalonamento através dos métodos de aproximação, são utilizadas, principalmente, as regras de prioridade e meta-heurísticas. Permitem encontrar soluções suficientemente boas em períodos de tempo útil.

A qualidade da resposta obtida através de um método de aproximação pode-se determinar se se conhecer previamente a solução óptima do problema e consequentemente calcular o desvio entre a solução encontrada e a solução óptima. Se a solução óptima não for conhecida ou é de difícil determinação, então assume-se a resolução suficientemente boa obtida através dos métodos de aproximação.

2.12. PROBLEMA DE MÁQUINA ÚNICA

O problema de sequenciamento de máquina única, em que *n* tarefas têm de ser sequenciadas é um problema elementar de escalonamento, a ordenação das tarefas determina completamente o plano de escalonamento. Considerando um ambiente de escalonamento estático, todas as tarefas estão disponíveis no início do processo, os tempos de processamento e datas de entrega de todas as tarefas que fazem parte do sistema são conhecidos e não dependem da sequência escolhida para a execução.

O modelo mais elementar de escalonamento em ambiente de máquina única verifica-se quando um conjunto de tarefas está disponível simultaneamente num só recurso ou máquina, num contexto determinístico. O problema de máquina única está limitado as seguintes condições (Baker, 2009), (Madureira, 2003):

- Existem *n* tarefas disponíveis simultaneamente para processamento no instante *t=0;*
- As máquinas, também denominadas como recursos, podem processar apenas uma tarefa de cada vez;
- Os tempos de preparação das tarefas são independentes da sequência escolhida e estão incluídos nos tempos de processamento;

- As características das tarefas são determinísticas e portanto conhecidas à partida;
- As máquinas nunca estão em espera enquanto há tarefas para escalonar;
- As máquinas estão constantemente disponíveis, não ocorrem avarias;
- Uma vez que a execução de uma tarefa começou, procede a sua execução de forma ininterrupta.

Debaixo destas condições existe uma correspondência um-a-um entre a sequência das *n* tarefas que devem ser executadas e a permutação do índice das mesmas. O total das combinações possíveis para o problema de máquina única é portanto *n*!. A título de exemplo, para uma instância de problema com 5 tarefas para escalonar, o total de combinações possíveis ascende a 120. Para 20 tarefas é de 2432902008176640000. O que torna impossível encontrar uma solução óptima sem ter de recorrer a técnicas de optimização.

Um caso clássico do problema de máquina única é aquele onde existem *n* tarefas a ser escalonadas e objectivo é reduzir o atraso máximo do sistema. Cada tarefa *j* tem um tempo de processamento p_j associado, assim como uma data de entrega d_j associada. De acordo com as premissas antes mencionadas e assumidas para este tipo de problema, as tarefas são executadas sem interrupção nem existem tempos mortos entre elas com a primeira tarefa a começar no instante *t=0*, torna-se evidente que para qualquer sequência assumida para a execução das tarefas tem um instante de conclusão C_j e um atraso associado $L_j = C_j - d_j$ Não sendo possível garantir a entrega de todas as tarefas antes, ou no limite, do prazo correspondente de entrega então o objectivo passa por reduzir o atraso associado à sequência de execução das tarefas. Dentro do conjunto de soluções possíveis poderá existir mais do que uma sequência com o mesmo atraso, pelo que o algoritmo terá de reflectir todas as restrições que sejam colocadas pelo sistema em causa (Pinedo, 2012), (Madureira, 2003).

Embora esta classe de problemas de escalonamento seja a mais elementar, é de muita importância no estudo dos problemas de escalonamento. O problema de máquina única é um caso especial de outros ambientes de produção, a análise de problemas inseridos neste contexto permite uma boa compreensão na análise e modelação de problemas de escalonamento mais complexos. Na prática, problemas de escalonamentos mais complexos são decompostos em sub-problemas, que permitem dividir o problema mais complexo em problemas de máquina única. Nos problemas em sistemas com várias máquinas pode

existir uma que compromete a capacidade de processamento de todo o sistema por apresentar uma capacidade de produção inferior à necessária, este tipo de situações, habitualmente designado por gargalo, pode conduzir a uma análise bastante complexa se for tratado de uma forma global, pelo que uma abordagem e divisão em sub-problemas de máquina única poderá facilitar e assim permitir a identificação e resolução do problema global.

É importante salientar que muitos problemas de máquina única pertencem à classe dos problemas de optimização de difícil resolução, com complexidade não polinomial, pelo que não existe nenhum algoritmo eficiente para sua resolução em tempo útil. Recorre-se a meta-heurísticas para a sua resolução, algumas delas serão apresentadas no capítulo 4.

2.13. RESUMO DO CAPÍTULO

Neste capítulo foram apresentados vários conceitos importantes no âmbito deste trabalho, relacionados com o escalonamento da produção. Na indústria existem vários ambientes de produção, dos quais resultam diferentes problemas de escalonamento, a sua classificação depende de vários critérios, alguns dos quais mencionados ao longo desta secção. Os problemas de escalonamento são classificados respeitando alguns critérios, nomeadamente a complexidade, a variabilidade dos parâmetros, o ambiente de produção onde estão inseridos, entre outros. Também foram descritas as diferenças entre os modelos de escalonamento teóricos e os modelos de escalonamento reais.

Neste capítulo também foi introduzido o conceito de complexidade computacional, importante na resolução dos problemas de escalonamento. Foram apresentadas algumas técnicas utilizadas na resolução dos problemas de escalonamento, que serão aprofundadas mais a frente.

Finalmente, foi apresentado o problema de máquina única, que representa o modelo mais elementar de escalonamento. Em ambientes reais muitos dos problemas de escalonamento são reduzidos a problemas mais simples, são classificados em sub-problemas de máquina única com o propósito de facilitar a análise de um problema mais complexo.

3. SISTEMAS DE ESCALONAMENTO

3.1. INTRODUÇÃO

A dificuldade dos problemas de escalonamento aumenta exponencialmente conforme o número de máquinas presentes no sistema. Para um problema com *n* máquinas, existem *n!* soluções possíveis. Para um problema com 3 máquinas existem 6 soluções possíveis. Para um com 5 máquinas existem 120, pelo que a necessidade de desenvolver sistemas informáticos capazes de lidar com as instâncias dos problemas de escalonamento tornou-se evidente.

Os sistemas desenvolvidos ao longo do tempo foram projectados para a indústria e para a utilização académica. São dotados de uma ampla variedade de configurações, possuem vários tipo de sistemas configuráveis e podem ser adaptados, na sua maioria, às necessidades particulares de cada utilizador. Apesar dos sistemas desenvolvidos serem genéricos na sua maioria, em grande parte dos sistemas podem ser especificados para um determinado sector da actividade em estudo ou no contexto particular de produção. Isto acontece devido a que os sistemas genéricos podem não se adaptar as características

particulares de determinado sistema e torna-se necessário o desenvolvimento de um sistema específico.

Actualmente existem empresas dedicadas ao desenvolvimento de *software* que servem de apoio para os sistemas de informação interno na indústria. A complexidade do código do *software* depende das características e do propósito para o qual foi desenvolvido. Se o objectivo for incluí-lo no sistema de informação central da empresa então o sistema tem de ser capaz de receber informação de outros sistemas de alto nível incluídos e actuar em função da informação que recebe constantemente. Por exemplo, pode interagir com um sistema que implemente o MRP (*Material Requirements Planning)* com a finalidade de lançar as datas de forma acertada para execução de tarefas (Pinedo, 2012).

Um sistema de escalonamento tem de gerar soluções robustas que permitam a sua adaptação à ocorrência de eventos que inviabilizem o plano original. Em problemas académicos a situação em que a instância é projectada é ideal, não existem paragens devido a avaria de máquinas, ou não aparecem tarefas que tem de ser inseridas no sistema de forma repentina. No entanto na indústria isto não acontece, pelo que num processo "reactivo" o plano original tem de ser alterado ajustando os objectivos sem comprometer o resultado final (Pinedo, 2012).

Geralmente o "motor" de um sistema de escalonamento é constituído por uma biblioteca de algoritmos disponíveis, que podem incluir por exemplo: Regras de prioridade, métodos de Pesquisa Local, técnicas baseadas no *Shifting Bottelneck,* técnicas baseadas no *branch – and-bound,*etc. A técnica escolhida para a resolução de um problema de escalonamento depende sempre das características do problema, da capacidade da máquina que processa o problema e do tempo disponível de espera para encontrar uma solução (Pinedo, 2012).

A interface do sistema é de extrema importância na sua utilização. É normalmente um factor eliminatório na escolha do *software* escolhido, em caso do sistema de escalonamento operar em PC's a interface é composta por múltiplas janelas, devido à preferência dos utilizadores em ter várias informações ao seu dispor de forma simultânea. Alguns sistemas permitem a modificação do estado actual do processo de escalonamento ou da informação do sistema, enquanto outros podem não permitir nenhuma modificação.

Os sistemas de escalonamento podem-se classificar em sistemas tradicionais e sistemas web.

3.2. SISTEMAS TRADICIONAIS

Consideram-se sistemas tradicionais aqueles que recorrem a tecnologias convencionais e que são utilizados na actualidade como apoio na tomada de decisão. Estão também presentes em ambientes académicos na introdução aos problemas de escalonamento e a utilização de técnicas adequadas no contexto onde se encontra inserida a instância a tratar. De entre os sistemas disponíveis serão apresentados o Lekin® e LISA®.

3.2.1. LEKIN SCHEDULING SYSTEM

Foi desenvolvido com o intuito de testar diferentes algoritmos e heurísticas na resolução de problemas de escalonamento. Esta versatilidade permite ao utilizador comparar o desempenho de heurísticas desenvolvidas face aos resultados obtidos mediante a aplicação das soluções apresentadas pelo motor do Lekin.

O Lekin permite simular uma variedade de ambientes, nomeadamente:

- Máquina única;
- Máquinas paralelas;
- Linha de fabrico;
- Oficina de fabrico;
- Linha de fabrico flexível;
- Oficina de fabrico flexível.

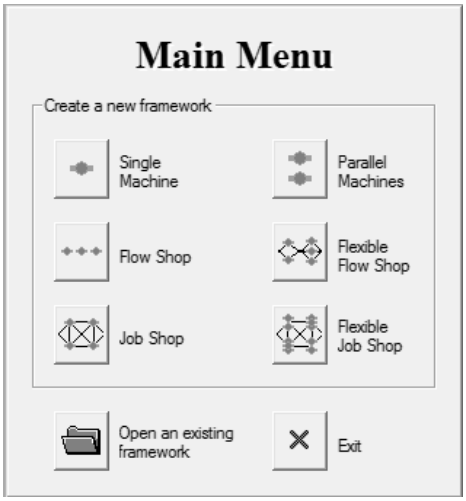


Figura 14 - Ambientes disponíveis no Lekin

A janela apresentada na Figura 14, é a primeira que aparece quando se inicia o Lekin e permite escolher o ambiente a simular. Os parâmetros solicitados pelo programa para configurar o sistema são o número de máquinas ou *workcenters* e o número de tarefas a executar, excepto para o caso de máquina única, que como é evidente, apenas utiliza uma máquina que executa *n* tarefas.

Para o contexto deste trabalho, foi escolhido o ambiente de máquina única com 5 tarefas a executar. Aparece uma janela na qual, entre outras coisas, é possível inserir uma identificação para cada uma das tarefas, data de lançamento, tempo de processamento, data de entrega e a penalização por tempo de atraso.

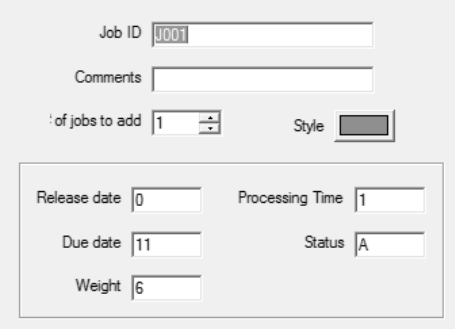


Figura 15 - Parâmetros solicitados

Após inserir todos os dados correspondentes à instância em análise na barra superior é possível escolher as alternativas para obter uma solução para o problema. No Lekin pode-se escolher entre as regras de prioridade e o algoritmo "*shifting-bottelneck*". Existem outros algoritmos no sistema, no entanto na versão utilizada não se encontram disponíveis.

Tabela 2 - Problema de escalonamento

Trabalho	Tempo de processamento	Data de entrega	Penalização
1	2	5	1
2	4	7	6
3	1	11	2
4	3	9	3
5	3	8	2

Na Tabela 2 é representado o problema de escalonamento utilizado como exemplo no Lekin. É um problema do tipo atraso pesado (*"weighted tardiness"*). Não existe data de lançamento nem existem tempos de transição entre tarefas. Para a resolução no Lekin foi

escolhida a heurística "*Shifting Bottleneck / sum wT"* por ser a mais adequada, dentro das disponíveis, para este tipo de problema.

Após o sistema correr o algoritmo é apresentada uma janela com o resultado. É possível, interagindo com as diferentes opções disponíveis, verificar a robustez da resolução e do algoritmo assim como visualizar a sequência encontrada pelo algoritmo. O utilizador consegue verificar através de uma janela os dados correspondentes ao desempenho da solução encontrada pelo sistema. É possível observar, entre outros, o atraso máximo, o tempo de percurso, o atraso total, entre outras.

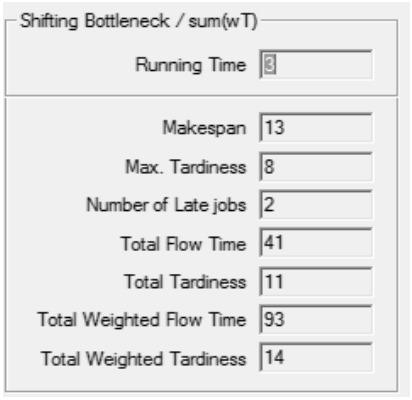


Figura 16 - Desempenho da solução

É possível também observar gráficos correspondentes à solução encontrada. A sequência encontrada pelo algoritmo é mostrada num gráfico de Gantt.

Em suma, o Lekin apresenta uma gama de possibilidades para obter uma solução a problemas de escalonamento. A escolha do sistema e posterior introdução dos dados é bastante intuitiva e de fácil percepção. A apresentação da resposta permite a realização de uma análise detalhada ao desempenho da mesma, disponibilizando diferentes formas de visualização da solução. O utilizador consegue guardar as soluções encontradas para comparar com outras obtidas através da execução de outros métodos ou heurísticas. É possível também importar algoritmos criados pelo utilizador que podem ser testados face às alternativas oferecidas pelo Lekin.

3.2.2. LISA, *A LIBRARY OF SCHEDULING ALGORITHMS*

LISA é um sistema desenvolvido para a resolução de problemas de escalonamento do tipo determinístico (todos os parâmetros são conhecidos previamente e fixos), nomeadamente problemas de máquina única e problemas do tipo oficina (linha de fabrico, oficina de fabrico, etc.). Neste contexto, assume-se que em cada instante de tempo que cada trabalho é processado numa máquina e cada máquina processa um trabalho (Andersen et al., 2010).

LISA pode ser utilizado para determinar a complexidade de um problema, para visualizar e manipular soluções obtidas através dos algoritmos e também pode servir como base para o desenvolvimento de algoritmos para problemas específicos (Andersen et al., 2010).

Existe uma vasta gama de algoritmos e heurísticas disponíveis no repertório do LISA para a resolução de problemas de escalonamento, a escolha, como é evidente, depende das características do problema a tratar. Nos métodos exactos apenas está disponível um algoritmo baseado em *branch-and-bound*. A sua utilização está limitada a um número reduzido de tarefas e máquinas devido ao tempo de execução do algoritmo. Nos métodos interactivos constam as regras de prioridade, algoritmos baseados no *shifting bottelneck*, e *beam-search.* Nos métodos de aproximação existem o *treshold accepting, simulated annealing*, os algoritmos genéticos, a pesquisa-tabu e os procedimentos baseado na colonia de formigas (Andersen et al., 2010).

A sequência do escalonamento é visível através de um gráfico de Gantt, como se pode ver na Figura 17. No exemplo é possível observar um plano escalonamento de 4 tarefas em 4 máquinas diferentes.

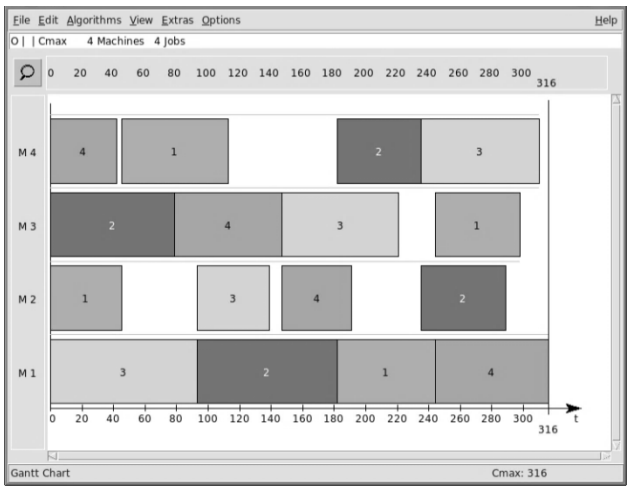


Figura 17 - Gráfico de Gantt no LISA (Andersen et al., 2010)

Existem outros sistemas de escalonamento disponíveis para além dos dois aqui mencionados. Na bibliografia é possível encontrar outros exemplos, tais como: *Real Time Dispatching and Agent Scheduling, Asprova Advanced Planning and Scheduling, Preactor Planning and Scheduling Systems, Taylor Scheduling Software.* No entanto o objectivo é comum: Proporcionar soluções para o complexo problema do Escalonamento. (Pinedo, 2012)

3.3. SISTEMAS WEB

Surgem como uma alternativa aos sistemas tradicionais. Apresentam a vantagem de poderem ser partilhados em rede. Hoje em dia é uma mais-valia, devido ao surgimento dos dispositivos móveis. Qualquer alteração realizada pode ser partilhada, sincronizando a informação nos diferentes dispositivos. A troca de informação de forma eficiente entre diferentes sectores ajuda na resolução de problemas de escalonamento. Outra característica dos sistemas web é que podem ser personalizados por forma a satisfazer necessidades individuais (Costa, 2014) (Pinedo, 2012). Não foi possível testar os sistemas, pelo que apenas será realizado um pequeno resumo de alguns exemplos disponíveis.

3.3.1. LEKINET

É um protótipo de um sistema de escalonamento web que surge como uma evolução do Lekin. É um sistema orientado para a resolução de sistemas de oficina de fabrico flexível e que valoriza os custos económicos associados à resolução de problemas. O objectivo é criar um sistema colaborativo formado por um grupo de agentes de escalonamento que partilhe informações entre si facilitando a resolução de problemas de escalonamento. Neste contexto cada sistema é denominado por agente (URL 5).

3.3.2. NEOS SERVER

É um sistema web que permite resolver problemas de optimização. O serviço encontra-se armazenado em servidores distribuídos pelo mundo. Um dos servidores está localizado na Universidade do Minho. Este serviço oferece mais de 60 programas para a resolução de problemas de optimização. Informação detalhada sobre os algoritmos disponíveis nos programas assim como uma guia com informação detalhada sobre o funcionamento do sistema pode ser encontrada em http://neos-guide.org/.

3.3.3. FORTMP

É sistema web de programação desenvolvido para resolver problemas de optimização de larga escala. Foi desenvolvido para resolver problemas de programação linear, programação quadrática, e problemas de programação inteira. Para a resolução de problemas de programação linear e programação quadrática recorre ao algoritmo Simplex primal e dual. Para a resolução de problemas de programação inteira recorre ao algoritmo *branch-na-bound* (URL 7).

3.4. RESUMO DO CAPÍTULO

O estudo computacional de problemas de escalonamento tem evoluído ao longo do tempo. A indústria também tem crescido pelo que se torna necessária que a evolução tecnológica acompanhe este crescimento. O surgimento de sistemas de escalonamento, cada um com a sua abordagem particular, tem ajudado a mitigar o aumento da dimensão dos problemas de escalonamento, na medida em que têm conseguido acompanhar esta evolução ao longo do tempo.

A interfase visual nos sistemas é uma das questões em constante evolução, já que torna-se indispensável que o aspecto visual dos sistemas seja apelável ao utilizador e é em muitos casos um factor decisivo na escolha.

Os sistemas de escalonamento surgem na versão tradicional e versão web. Actualmente, num mundo dominado pela internet e a tecnologia em geral, os sistemas web apresentam a capacidade de conseguir partilhar toda a informação relativa ao escalonamento na rede e sincronizar a mesma com todos os agentes participantes no escalonamento.

Os algoritmos presentes nos sistemas podem ser genéricos ou mais específicos. Desenvolver um sistema de raiz apresenta a vantagem de poder adaptar-se às características particulares do ambiente onde o sistema vai ser inserido, assim como aos gostos particulares de cada utilizador.

A utilização dos sistemas de escalonamento representa uma ajuda preciosa no apoio à tomada de decisão. Recorrer a um sistema permite coordenar melhor as tarefas que devem ser executadas pelas máquinas presentes num determinado ambiente para o cumprimento

dos prazos exigidos, o que nem sempre é fácil de conseguir, devido aos constrangimentos que surgem nos ambientes industriais (avarias, paragens inesperadas, etc.)

4. MÉTODOS DE OPTIMIZAÇÃO

4.1. INTRODUÇÃO

A complexidade de um problema do escalonamento depende dos factores que envolvem o sistema onde vai ser inserido. Encontrar uma solução óptima para um problema de escalonamento nem sempre é possível, devido às restrições económicas e computacionais.

Existem problemas de escalonamento de pequena dimensão, cuja solução é fácil de encontrar num período de tempo útil. Podem ser formulados como problemas de programação linear e podem ser aplicados algoritmos que permitem uma resolução em tempo útil e de forma eficiente. A dificuldade cresce de forma polinomial e não de forma exponencial (Madureira, 2003), (Pinedo, 2009).

À medida que a dimensão do problema aumenta a sua complexidade também aumenta de forma exponencial. Neste caso, não existem algoritmos ou regras que possam ser aplicadas de forma a encontrar uma solução óptima eficiente e em tempo útil. Estes tipos de problemas de optimização, classificados como NP-difíceis, não podem ser formulados como um problema de programação linear. Resolver estes problemas até a optimização pode exigir um enorme esforço computacional. Pelo que, na prática, considera-se uma

solução satisfatória, conseguida em tempo útil. O desvio da solução encontrada face à solução óptima só se consegue determinar se a solução óptima for previamente conhecida, o que nem sempre é assim (Madureira, 2003), (Pinedo, 2009).

Os métodos para resolução de problemas optimização podem ser classificados como **métodos exactos** e **métodos aproximados**. No caso dos métodos exactos o objectivo é encontrar a solução óptima para o problema recorrendo para isso a uma pesquisa exaustiva do espaço solução. Este tipo de métodos é utilizado em problemas de pequena dimensão. Os métodos de aproximação têm como objectivo encontrar soluções satisfatórias para o problema em tempo útil. A utilização dos métodos de aproximação não garante a solução óptima mas sim soluções muito próximas da solução óptima. Os métodos de aproximação podem ser aplicados a problemas de grande dimensão, uma vez que são eficientes em termos de tempo.

4.2. MÉTODOS EXACTOS

Exploram as soluções possíveis dentro do espaço de um problema no sentido de encontrar a melhor solução possível. Isto é, a solução óptima para o problema respeitando as restrições colocadas para a formulação do mesmo e os critérios de optimização exigidos. Para problemas de elevado grau de dificuldade, NP-difíceis, a complexidade cresce exponencialmente com a dimensão do problema, pelo que estes algoritmos não conseguem encontrar uma solução óptima num intervalo de tempo útil, requerendo também um grande esforço computacional. Por esta razão, os métodos exactos devem ser aplicados em problemas de pequena dimensão, nos quais apresentam um bom desempenho. Alguns dos métodos exactos utilizados para resolver problemas de escalonamento são o *"Branch and Bound"* e a programação dinâmica (Pinedo, 2009) (Madureira, 2003).

4.2.1. MÉTODO *BRANCH AND BOUND*

A ideia básica de um algoritmo baseado no *branch and bound* consiste na divisão do espaço da solução (*branching),* cada divisão do espaço é considerada de forma separada. Os limites (*bound)* da função objectivo são inferiores se o objectivo for o de minimizar ou superiores se o objectivo for maximizar. As ramificações são exploradas de forma exaustiva, procurando a solução óptima para o problema em análise, sempre respeitando os

limites estabelecidos e aquelas ramificações que apresentarem soluções piores da encontrada até a data são descartadas (Baker, 2009).

O sucesso do algoritmo depende da eficiência com a qual forem determinados os limites superiores ou inferiores do espaço da resposta. É o método mais comum na resolução de instâncias de programação linear inteira. O objectivo de utilizar o *branch and bound* é precisamente o de encontrar uma solução óptima inteira. A ramificação do problema principal implica o surgimento de sub-problemas que são (Baker, 2009):

- Sub-problemas mutuamente exclusivos e exaustivos do original;
- Versões do problema original, parcialmente resolvidas;
- Problemas mais pequenos que o original.

O problema principal é ramificado em duas ou mais partições. Depois cada uma das ramificações é analisada exaustivamente até ser encontrada uma solução que satisfaça as restrições do problema e a função objectivo. Todas as ramificações que apresentarem um valor que piore a melhor solução encontrada são descartadas, a razão é simples: As soluções encontradas em cada ramificação representam um limite superior ou inferior, conforme o objectivo for maximizar ou minimizar. Ou seja, nenhuma solução resultante da ramificação vai melhorar o valor encontrado e por isso são descartadas.

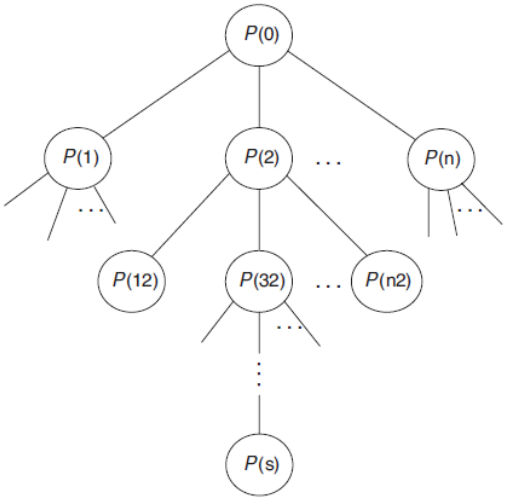


Figura 18 - Exemplo de *branch and bound* (Baker, 2009)

O exemplo descrito, na Figura 18, representa um problema de sequenciamento em ambiente de máquina única. O problema original é representando por P(0). Como se pode observar na figura é divido em sub-problemas P(1), P(2)…. P(n). A quantidade de ramificações, neste caso particular, depende da quantidade de tarefas que no contexto deste

problema é "n". P(1) representa o mesmo problema, mas com a tarefa 1 fixada na última posição, P(2) representa a tarefa 2 fixada na última posição, sendo a lógica a mesma para *n* tarefas. A dimensão dos sub-problemas é claramente menor que o problema P(0) porque apenas existem *n-1* tarefas por sequenciar e cada ramificação P(*i*) representa uma solução parcial do problema P(0). Cada ramificação pode também ser dividida em sub-problemas, P(1,2), P(3,2)…. P(n,2). Na solução parcial P(1,2) as tarefas 1 e 2 ocupam os últimos dois lugares no sequenciamento, mantendo esta lógica para os outros sub-problemas (Baker, 2009).

Na escolha do método *branch and bound* parte-se de uma solução, não inteira, dividindo o problema em ramificações. Cada ramificação representa uma solução parcial do problema original. No entanto há que ter em consideração que a árvore de sub-problemas pode crescer tornando o processo algo demorado. Para ultrapassar isto podem-se escolher métodos de paragem para o algoritmo que executa o *branch and bound*, no entanto há que tomar em consideração que se pode estar a comprometer uma solução melhor do que a actual, uma consequência que tem de se assumir se o tempo disponível para encontrar uma solução ao problema em análise for limitado.

4.2.2. PROGRAMAÇÃO DINÂMICA

É uma técnica aplicada na resolução de problemas de optimização. Consiste na divisão do problema principal em sub-problemas. A semelhança do *branch and bound* o problema principal é dividido em sub-problemas resolvendo cada um até encontrar uma solução óptima para o problema original. Cada sub-problema representa uma solução parcial do principal. Em teoria a divisão simplifica o problema original e reduz o esforço computacional necessário para a resolução do problema de optimização (Pinedo, 2009).

Dividir um problema de optimização em sub-problemas pode implicar o crescimento deste de forma exponencial. O crescimento da "árvore" vai depender da dimensão particular do problema. Um algoritmo baseado em programação dinâmica utiliza a informação obtida na resolução de outros sub-problemas e armazena. Cada vez que precisar dessa informação simplesmente faz uma pesquisa nas soluções já encontradas, reduzindo assim o esforço computacional aplicado na resolução de um problema (Pinedo, 2009) (Baker, 2009).

A programação dinâmica é caracterizada por 3 tipo de equações, nomeadamente (Pinedo, 2009):

- As condições iniciais do problema;
- Relação recursiva entre as variáveis
- Função objectivo para obtenção do valor óptimo.

É com base nessas 3 premissas que podem ser aplicados algoritmos de programação dinâmica a um problema de optimização.

Outros métodos e algoritmos assim como resultados obtidos mediante a aplicação de métodos exactos podem ser encontrados em (Papadimitrou,1998) (Shrijver, 2004), (Pinedo, 2009), (Baker, 2009).

Em problemas de pequena dimensão uma abordagem recorrendo aos métodos exactos pode conduzir à solução óptima. No entanto, a medida que a dimensão do problema aumenta e a sua dificuldade aumenta exponencialmente, outra abordagem é necessária.

4.3. MÉTODOS DE APROXIMAÇÃO

Esta classe de métodos é caracterizada por encontrar soluções satisfatórias num período de tempo aceitável. O tempo disponível para obter uma solução óptima a um determinado problema nem sempre é muito, pelo que se tem de fazer um compromisso e aceitar uma boa solução obtida em tempo útil (Madureira, 2003).

São aplicados quando para a instância não existem algoritmos passíveis de determinar a solução óptima em tempo útil e com recursos computacionais limitados. Recorrer aos métodos de aproximação implica encontrar soluções satisfatórias, em muitos dos casos perto da solução óptima. Estes métodos podem ser aplicados a problemas de grande dimensão (Madureira, 2003) (Pinedo, 2009).

Nesta secção serão apresentados alguns métodos de aproximação, com especial ênfase às meta-heurísticas.

4.3.1. REGRAS DE PRIORIDADE

Na prática são frequentemente utilizados procedimentos simples de sequenciamento designados como regras de prioridade. Estas regras são aplicadas para determinar a sequência, segundo a qual as operações deverão ser executadas, considerando os critérios de desempenho exigidos ao sistema.

O esquema de sequenciamento e respectiva priorização das tarefas é realizado considerando atributos das próprias tarefas, dos atributos das máquinas e do tempo actual em que se encontra o processo de produção. O funcionamento é simples, sempre que uma máquina está livre, a regra de prioridade escolhe uma das tarefas, dentro do lote ainda em espera de ser sequenciado e de acordo com o critério escolhido, a tarefa com a prioridade mais alta (Madureira, 2003) (Pinedo, 2009).

A classificação das regras de prioridade pode ser feita de diferentes formas. De acordo com a variabilidade de informação (Pinedo, 2009):

- **Estáticas -** As regras enquadradas neste contexto não dependem da evolução do tempo, são fixas. Partem do princípio de que o conjunto de tarefas a sequenciar não varia até o processamento das mesmas estar concluído. Exemplo destas regras são aquelas baseadas na data de entrega ou data de lançamento.
- **Dinâmicas –** As regras enquadradas neste contexto dependem da evolução temporal. São regras aplicadas em tempo real, a medida que os eventos acontecem. A prioridade de processamento das tarefas pode mudar ao longo do sequenciamento.

Outra forma de classificar as regras de prioridade é de acordo com o tipo de informação em que a regra se baseia (Pinedo, 2009).

- **Local –** Utiliza apenas informação relativa à fila onde a tarefa está em espera ou à máquina da fila onde se encontra a tarefa.
- **Global –** Consideram informação relativa a outras máquinas dentro do processo produtivo. Por exemplo: Informação relativa ao tempo de processamento da tarefa da próxima máquina no percurso ou informação relativa ao cumprimento da fila de espera dessa máquina.

Algumas das regras de prioridade mais utilizadas são:

Tabela 3 - Regras de prioridade

	Regra	Definição
Regras dependentes das datas de entrega e data de lançamento	"EDD" - Earliest due date	É dada prioridade a tarefas com data de entrega mais cedo
	"ERD" – Earliest release date	É dada prioridade a tarefas com data de lançamento mais cedo
	"MS" – Minimum Slack	É uma regra dinâmica. Calcula a folga mínima e é dada a prioridade ao trabalho com o valor mais baixo.
Regras dependentes dos tempos de processamentos	"LPT" – Longest processing time	É dada prioridade à tarefa com o maior tempo de processamento
	"SPT" – Shortest processing time	É dada prioridade à tarefa com o menor tempo de processamento
	"WSPT" – Weighted shortest processing time	É dada a prioridade à tarefa com o maior rácio Wj/Pj
	"CP" – Critical path	Selecciona como próxima tarefa a primeira da lista de tempos de processamento, de acordo o gráfico de precedências
	"LNS" - Largest number of successors	Selecciona como próxima tarefa a que apresentar o maior número de sucessores no gráfico de precedências.
Outras regras	"SIRO" – Service in random number	É seleccionada uma tarefa de forma aleatória
	"SST" – Shortest Setup time first	É seleccionada a tarefa com o menor tempo de preparação
	"LFJ" – Least flexible job first	É seleccionada a tarefa a ser processada pelo menor número de máquinas
	"SQNO" – Shortest queue at the next operator	É seleccionada a tarefa com a menor fila de espera na próxima máquina.

A escolha da regra de prioridade que será aplicada ao sistema de produção está dependente dos objectivos da empresa em relação ao sistema. Se o objectivo for reduzir o atraso máximo, então terão de ser aplicadas as regras relacionadas com as datas de entrega e lançamento. Com objectivos relacionados ao tempo de processamento, por exemplo reduzir o tempo de conclusão das tarefas, balanceamento das cargas de trabalho em ambientes de máquina paralela, entre outros, as regras que deverão ser aplicadas correspondem às dependentes dos tempos de processamento. Para reduzir o tempo de paragem das máquinas, a regra "SQNO" pode ser aplicada e se se pretender uma facilidade na implementação do sistema a regra "SIRO" pode ser a escolhida.

Estas regras de prioridade simples são úteis e apresentando resultados satisfatórios em termos de escalonamento quando o objectivo é apenas reduzir o "*makespan*" ou reduzir o atraso máximo. No entanto, na prática os objectivos são mais complicados. Em sistemas reais os objectivos da empresa em relação ao sistema de produção pode ser uma combinação de vários objectivos mais simples ou podem estar dependentes do tempo ou do número de tarefas a espera de ser processada. A aplicação de uma ou duas das regras de prioridade pode produzir um escalonamento inaceitável para a empresa. Neste contexto deverão ser consideradas regras de prioridade que considere uma quantidade de parâmetros e que podem lidar melhor com a natureza dos objectivos, apresentando melhores resultados. Estas regras são denominadas como **regras de prioridade compostas** e não são mais do que a combinação de algumas das regras de prioridade básicas antes mencionadas (Pinedo, 2009).

À medida que um determinado atributo afecta a prioridade global de uma tarefa é determinado com base a regra de prioridade simples que o utiliza e um parâmetro de escala. Cada uma das regras de prioridade simples que interage com as outras numa regra composta tem designado o seu próprio parâmetro de escala, que é escolhido para dimensionar correctamente a contribuição da regra básica na expressão final da regra composta. O parâmetro de escala pode ser fixo ou variável ou em função do conjunto de tarefas específicas que se pretendem escalonar. No caso da última é preciso realizar um esforço computacional no sentido de determinar algumas estatísticas que caracterizem a instância do problema com a maior precisão possível. Estas estatísticas são denominadas como "factores", normalmente não dependem do escalonamento e podem ser calculados

com base nas características de uma determinada tarefa ou dos atributos da máquina ou máquinas que pertencem ao sistema (Pinedo, 2009).

Cada vez que uma regra de prioridade composta é utilizada para o escalonamento, as estatísticas necessárias são calculadas. Os cálculos dos parâmetros de escala são estabelecidos por funções pré-determinadas, utilizando como base os valores estatísticos. Após o cálculo dos parâmetros de escala, a regra de prioridade composta é aplicada ao conjunto de tarefas.

Em Pinedo (2009) é apresentada uma regra de prioridade composta denominada por *"ATC" – Apparent Tardiness Cost.* Combina as regras de prioridade WSPT e MS. O objectivo desta regra passa por reduzir o atraso pesado do sistema. As tarefas escalonadas com esta regra são escolhidas de acordo com um índice, calculado sempre que a máquina está livre, a tarefa que apresentar o maior valor é a próxima a ser executada.

$$I_j(t) = \frac{w_j}{p_j} \times e^{\left(-\frac{\max(d_j - p_j - t, 0)}{K\bar{p}}\right)} \quad (14)$$

Na equação 14, w_j corresponde ao peso da tarefa j, p_j ao tempo de processamento da tarefa j, K a parâmetro de escala e $\bar{p}$ corresponde à média dos tempos de processamento restantes. O comportamento da equação depende do valor do parâmetro de escala. Conforme este for muito alto ou muito baixo, a regra pode ser reduzida para WSPT ou MS, respectivamente. Uma análise mais detalhada desta regra pode ser encontrada em Pinedo (2009).

As regras de prioridade apresentadas anteriormente podem ser classificadas como sendo métodos construtivos. O plano de escalonamento é realizado adicionando uma tarefa de cada vez, sendo construído de forma gradual.

4.3.2. ALGORITMO *SHIFTING BOTTELNECK*

É um método utilizado com o objectivo de reduzir o *´makespan´* de um sistema, nomeadamente em ambientes de oficina de fabrico (*Job-shop*). Foi introduzido por (Adams et al., 1988). Assumindo que num sistema existem tarefas que competem pelos mesmos recursos, inevitavelmente existe a possibilidade do sistema encontrar um gargalo. É esta situação que se tenta ultrapassar recorrendo a este método (Madureira, 2003), (Pinedo, 2012).

Este método resolve em cada iteração um problema de máquina única e o objectivo é tratar todas as máquinas que pertencem ao sistema. O problema pode ser divido em 3 etapas: a identificação dos sub-problemas, a selecção da máquina crítica ou gargalo e a solução do problema e re-optimização (Madureira, 2003).

O algoritmo começa por simplificar o problema original em *m* problemas de máquina única, resolvendo um de cada vez. A máquina com o maior '*makespan'* é considerada a máquina crítica, sendo esta a primeira máquina escalonada, por ser a máquina que provoca um estrangulamento no sistema. A máquina crítica é a primeira a ser escalonada, seguindo-se as restantes máquinas que fazem parte do sistema, recorrendo a um método iterativo (Madureira, 2003).

Depois de sequenciada a máquina crítica são resolvidos novos problemas de máquina única, recorrendo a métodos iterativos. Na bibliografia é referenciado o *branch-and-bound*, o método de Carlier, entre outros como os métodos utilizados neste processo. O processo de re-optimização é repetido até que todas as máquinas estejam sequenciadas. (Madureira 2003), (Pinedo, 2012).

4.3.3. META HEURÍSTICAS

A palavra Meta-heurística provém do grego *"heuriskein"* cujo significado é a arte de descobrir novas estratégias para resolução de problemas. O termo Meta-heurística foi introduzido por F.Glover em 1986 e são definidos como metodologias que podem ser utilizadas como estratégia na concepção de heurísticas subjacentes para a resolução de problemas de optimização específicos (El-Ghazali, 2009).

As Meta-heurísticas surgem como alternativa na resolução de problemas de optimização combinatória de difícil resolução (*NP-hard*). O esforço computacional necessário para a aplicação de uma técnica ou método de optimização tem de ser considerado na escolha do método. Tal como já foi referido anteriormente, neste tipo de problemas a complexidade cresce a medida que a dimensão do problema aumenta Quando o número de soluções possíveis é elevado o processo da procura exaustiva da solução que representa um melhor desempenho para o sistema é moroso. Muitas vezes esta demora é incompatível quando se exigem respostas quase imediatas para solucionar os problemas que surgem no processo. Às vezes tem de se fazer um compromisso entre a qualidade da resposta e o tempo

utilizado na procura da mesma, obtendo respostas satisfatórias para a resolução dos problemas em tempo útil (Madureira 2003), (Baker, 2009).

A utilização das Meta-Heurísticas garante uma solução de boa qualidade obtida de forma eficiente. Guiam-se por um processo iterativo que modifica operações de heurísticas subordinadas para produzir respostas de boa qualidade, manipulando uma solução ou conjunto de soluções em cada iteração até obedecer um critério de paragem ou não conseguir melhorar a solução obtida dentro dos parâmetros nos quais estão a ser executadas (Madureira, 2003), (El-Ghazali, 2009).

Nesta secção serão apresentadas Meta-Heurísticas baseadas em **Pesquisa Local,** que partem de uma solução inicial e pesquisam uma única solução, dentro do espaço de soluções, através de um processo iterativo até encontrar o óptimo global ou óptimo local. Também serão apresentadas Meta-Heurísticas baseadas em **Populações**, que partem de uma população inicial gerando de forma iterativa soluções que são integradas na população actual através de um mecanismo de selecção (Madureira, 2009), (El-Ghazali, 2009).

4.3.4. META-HEURÍSTICAS BASEADAS EM PESQUISA LOCAL

Partem de uma solução inicial, que pode ser obtida através de uma das regras de prioridade apresentadas, e através de sucessivas iterações procuram melhorar o objectivo proposto para o plano de escalonamento. Neste contexto, estão os algoritmos baseados em pesquisa local. Os problemas de escalonamento apresentam, na sua grande maioria, uma dificuldade computacional que não permite a obtenção de uma resposta óptima em tempo útil. Os algoritmos baseados na pesquisa local não garantem, à partida, a solução óptima para o problema em questão, no entanto a resposta obtida a partir da sua aplicação é satisfatória e pode ser obtida em tempo útil sem necessidade de recorrer a grandes esforços computacionais. São suficientemente flexíveis de modo a lidar com as restrições específicas dos problemas. (Madureira, 2003), (Pinedo, 2009).

Partindo da solução inicial, em cada iteração do algoritmo a solução actual é substituída por uma da vizinhança e que melhora a solução actual. O algoritmo termina quando na vizinhança actual não existir nenhuma solução que melhore a actual. Nesta situação atingiu-se um óptimo local (Madureira, 2009), (El-Ghazali, 2009).

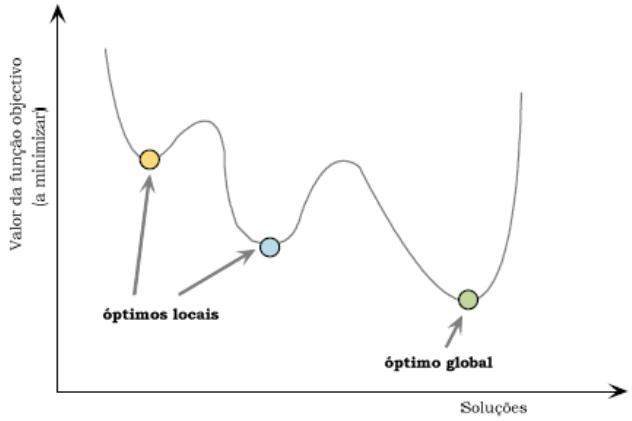


Figura 19 - Óptimos locais e globais (Madureira, 2009)

Neste tipo de situação estamos perante uma limitação do algoritmo. Dado que o facto de encontrar um óptimo local não implica a não existência de uma outra solução melhor do que a actual dentro do espaço da solução, como se pode verificar na Figura 19.

A estrutura da vizinhança assume portanto uma importância relevante nas meta-heurísticas baseadas em pesquisa local. Em ambientes de máquina única a vizinhança poderá ser criada a partir da troca de tarefas adjacentes. Em alguns casos pode ser criada trocando duas quaisquer tarefas com um afastamento definido, a trocar de três tarefas entre outros. (Madureira, 2003), (Pinedo, 2009).

4.3.4.1. SOLUÇÃO INICIAL

Gerar uma solução inicial pode ser feito de forma aleatória, na qual a ordem dos componentes na solução é obtida de forma aleatória. Também podem ser obtidas através de uma das regras de prioridade apresentadas anteriormente (Madureira, 2009).

4.3.4.2. VIZINHANÇA E SUB-VIZINHANÇAS

A estrutura de uma vizinhança é crucial em Meta-Heurísticas de Pesquisa Local. Se a estrutura não for a adequada ao problema, qualquer Meta-Heurística falhará na resolução do problema (El-Ghazali, 2009).

Pode ser definido como o conjunto de soluções a partir de do qual cada uma das soluções do problema é gerado. Considerando que S representa o conjunto de soluções admissíveis, a função N da vizinhança pode ser definida por $N : S \rightarrow 2^S$ (Madureira, 2009), (El-Ghazali, 2009).

Os mecanismos geradores de vizinhanças mais comuns no escalonamento estão baseados nas permutações. A vizinhança é obtida através da troca de duas tarefas adjacentes, duas quaisquer tarefas, a troca de três tarefas, entre outros. Há que referir no entanto que para poder realizar a troca de operações, tem de pertencer à mesma máquina e a diferentes tarefas (Madureira, 2009), (Pinedo,2009).

Figura 20 - Exemplo de vizinhança

Na Figura 20 é possível observar a troca de duas tarefas adjacentes como mecanismo de geração de vizinhança. Recorrendo a este mecanismo a dimensão da vizinhança é *n-1*. Em que *n* corresponde à dimensão do problema.

4.3.4.3. ALGORITMO DE PESQUISA LOCAL

O funcionamento básico de um algoritmo de pesquisa local consiste em partir de uma solução inicial, que pode ser gerada a partir de uma regra de prioridade. A seguir é gerada a vizinhança dessa solução, o objectivo do algoritmo é escolher a melhor solução dentro da vizinhança e fazer dela a solução principal. A forma mais simples de escolher uma nova solução consiste em seleccionar uma que melhore o valor da função objectivo O processo repete-se enquanto existir, dentro da vizinhança, uma solução que melhore a principal. Quando isto não ocorrer é porque o algoritmo atingiu um óptimo local e o processo acaba. Na Figura 21, é possível apreciar um algoritmo baseado na Pesquisa Local (Madureira, 2009).

Algoritmo 1: Algoritmo de Pesquisa Local

```
Início
    n←1
    Seleccionar x1 ∈ F
        x1 ← solução_inicial()
        x* ← x1
        f* ← f(x1)
    N(x1) ← gera_vizinhança()
    xn+1 ← solução_melhor(N(x1))
    Fazer
        Se f(xn+1) ≤ f(x*) então
            xn ← xn+1
            x* ← xn+1
            f* ← f(xn+1)
        FimSe
        n←n+1
        N(xn) ← gera_vizinhança()
        xn+1 ← solução_melhor(N(xn))
    Enquanto f(xn+1) ≤f(xn)
Fim
```

Figura 21 - Algoritmo de Pesquisa Local (Madureira, 2009)

Neste algoritmo a solução inicial é identificada por x_1, a função objectivo por $f(x_1)$, a vizinhança por $N(x_1)$ e a melhor solução, dentro da vizinhança, x_{n+1}.

4.3.4.4. PESQUISA TABU

A Pesquisa Tabu foi concebida com o intuito de escapar aos mínimos locais referidos no algoritmo de Pesquisa Local. A diferencia entre o algoritmo de Pesquisa Local e o de Pesquisa Tabu é a utilização do conceito de memória.

O método foi introduzido por Fred Glover em 1989. A ideai principal do algoritmo é de que é possível ultrapassar os mínimos locais aceitando uma degradação no valor da função objectivo que se está a considerar. As soluções anteriormente visitadas são armazenadas numa lista de soluções proibidas, lista tabu.

O funcionamento básico do método é idêntico ao algoritmo de Pesquisa Local, no sentido de o espaço de soluções é explorado deslocando-se de uma sequência para outra, escolhendo a melhor solução possível, no entanto algumas das iterações podem ser consideradas tabu na tentativa de escapar aos mínimos globais. Quando atinge óptimo local é seleccionada como melhor solução uma que piore a solução actual, continuando com o algoritmo, passando esta a ser a melhor solução actual. Existe a possibilidade de entrar num ciclo ao visitar soluções que já foram previamente visitadas, é por isso que as trocas de tarefas adjacentes na actual solução são inseridas na lista tabu, actualizando a lista em

cada iteração. O algoritmo tem verificar em cada iteração se a troca está proibida. A lista tem um número limite de entradas definidas, e em cada iteração esta lista é actualizada sendo que sempre que uma solução é visitada entra na lista e a mais antiga sai. Entrar num ciclo continua a ser uma possibilidade, é por isso que é definido um critério para a paragem do algoritmo (Madureira, 2009), (El-Ghazali, 2009), (Pinedo, 2009).

Na utilização de algoritmos baseados na pesquisa tabu o conceito de memória está presente nos mecanismos de intensificação e diversificação que permitem uma análise sistemática do espaço da solução através do registo do percurso recorrido pelo algoritmo em cada iteração.

Tabela 4 - Tipos de memória da Pesquisa Tabu (Madureira, 2009)

Memória da pesquisa	Função	Representação
Lista Tabu	Evita ciclos	Soluções visitadas, atributos dos movimentos, soluções dos atributos
Memória de médio prazo	Intensificação	Memória recente
Memória de longo prazo	Diversificação	Memória frequente

Na Tabela 4, é possível verificar os diferentes tipos de memória que compõem a Pesquisa Tabu. O uso de memória vai influenciar o processo de pesquisa da solução óptima, guiando-a com base nas soluções previamente visitadas.

- A função a curto prazo consiste em armazenar o histórico de soluções visitadas, para evitar cair em ciclos (Madureira, 2009), (El-Ghazali, 2009);
- A memória a médio prazo consiste na exploração da informação das melhores soluções encontradas para guiar a pesquisa a regiões promissoras do espaço de pesquisa. A intensificação consiste na definição do critério de escolha da solução seguinte com base na informação recolhida das melhores soluções encontradas;
- A memória de longo prazo consiste na encorajar a diversificação da pesquisa, obrigando a exploração em regiões do espaço ainda não exploradas .

4.3.4.4.1. PARAMETRIZAÇÃO DO ALGORITMO DA PESQUISA TABU

Tal como acontece no algoritmo de pesquisa local, na Pesquisa Tabu a estrutura da vizinhança é importante. Os métodos para gerar vizinhanças anteriormente também podem ser aplicados na Pesquisa Tabu. Devido às características particulares do algoritmo também tem de ser tomadas algumas decisões, nomeadamente a escolha dos atributos a serem guardados na lista tabu, a definição do tamanho e a selecção do critério de paragem.

- **Escolha dos atributos dos movimentos a serem guardados na lista tabu:** Armazenar todas as soluções previamente visitadas não é uma forma viável de constituir a lista tabu. Existe a necessidade de testar para cada movimento candidato no espaço da vizinhança a existência de algum movimento inverso já presente na lista. Uma alternativa na lista tabu é considerar uma ou mais características do movimento e não o par de soluções. Por exemplo se pode considerar a transformação realizada na solução actual.

  Existe no entanto uma consequência ao proceder desta forma. Por exemplo, se o movimento de uma solução para outra é obtido ao trocar as tarefas 2 e 3 de posição, é inserido na lista tabi. Isto leva à exclusão de muitas mais soluções do que aquela que foi visitada.

  Para ultrapassar este problema, na pesquisa tabu existe um mecanismo denominado por critério de aspiração. Estabelece o que é uma solução suficientemente boa e permite que uma solução gerada a partir de uma transformação tabu seja escolhida se esta melhorar o valor da função objectivo (Madureira 2009), (El-Ghazali, 2009).
- **Definição do tamanho da Lista Tabu:** Corresponde ao número máximo de atributos que vão ser armazenados na lista. Na bibliografia é referido que normalmente são guardados entre 7 e 9 atributos. O tamanho da lista pode ser definido de forma estática (valor fixo) ou de forma dinâmica (o tamanho pode variar durante a pesquisa) (Madureira 2009), (El-Ghazali, 2009).
- **Selecção do critério de paragem:** Para o sucesso do algoritmo e por forma a reduzir o tempo e poupar algum esforço computacional é definido um critério de paragem para o algoritmo. Atingir um número pré-determinado de iterações ou definir um critério relacionado com a qualidade de solução são alguns dos mais frequentes (Madureira 2009), (El-Ghazali, 2009).

4.3.4.4.2. ALGORITMO BASEADO NA PESQUISA TABU

```
Início
    n←1
    x1 ←solução_inicial()
    Fmelhor ←F(x1)
    xmelhor ←x1
    Iniciar_ListaTabu()
    N(xn) ← gera_vizinhança()
    Enquanto criterio_paragem = falso fazer
      F* ←∞
      Para todos x ∈ N(xn)
       Se F(x) < F* então
            m ←modificação (xn, x)
            Se m ∉ ListaTabu  ou  (m∈Lista_Tabu e F(x) <Fmelhor) então
               F* ←F(x)
               x*  ←x
               m* ←m
            FimSe
       FimSe
      FimPara
      xn+1 ← x*
      inserir_ListaTabu(m*)
      Se  F* < Fmelhor então
        xmelhor ← x*
        Fmelhor ← F*
      FimSe
    n←n+1
    FimEnquanto
Fim
```

Figura 22 - Algoritmo baseado na Pesquisa Tabu (Madureira, 2009)

O funcionamento do algoritmo segue a lógica previamente explicada. Parte-se de uma solução inicial e é estabelecido um critério de paragem para o algoritmo. Este vai percorrer o espaço de pesquisa gerado através do conceito de vizinhança adicionando elementos na lista tabu conforme se deslocar de uma solução para outra e pára quando for atingido o critério de paragem.

4.3.4.4.3. EXEMPLO ILUSTRATIVO

Na Figura 23 é mostrado um exemplo do algoritmo de pesquisa tabu para um problema de máquina simples.

Iteração	Solução x_n	VizinhançaN (x_n)	Valor de f	Lista Tabu
1	5 3 2 4 1	3 5 2 4 1 **5 2 3 4 1** 5 3 4 2 1 5 3 2 1 4	20 **14** 32 23	3 → 2
2	5 2 3 4 1	2 5 3 4 1 5 3 2 4 1 **5 2 4 3 1** 5 2 3 1 4	14 20 **11** 17	3 → 4 3 → 2
3	5 2 4 3 1	**2 5 4 3 1** 5 4 2 3 1 5 2 3 4 1 5 2 4 1 3	**11** 26 14 14	5 → 2 3 → 4
4	2 5 4 3 1	5 2 4 3 1 **2 4 5 3 1** 2 5 3 4 1 2 5 4 1 3	11 **12** 14 14	5 → 4 5 → 2
5	2 4 5 3 1	**4 2 5 3 1** 2 5 4 3 1 2 4 3 5 1 2 4 5 1 3	**12** 11 14 15	2 → 4 5 → 4

Figura 23 - Exemplo ilustrativo (Madureira, 2009)

No exemplo o critério de paragem para o algoritmo foi definido em 5 iterações. Na primeira iteração parte-se solução inicial $x_1 = [5\ 3\ 2\ 4\ 1]$ e dentro da vizinhança dessa solução, gerada a partir da troca de tarefas adjacentes, é possível identificar uma solução que melhora o valor da função objectivo trocando de posição as tarefas 3 e 2. Essa troca é adicionada à lista tabu, cujo tamanho é constituído por dois elementos, e na seguinte iteração, as soluções encontradas visando a troca das tarefas 3 e 2 não são exploradas.

4.3.4.5. *SIMULATED ANNEALING*

Foi introduzido por Kirkpatrick et al., 1983 e Cerny, 1985 e foi inspirado na termodinâmica e na metalurgia. A base do algoritmo é assente no processo mediante o qual o metal arrefece lentamente até solidificar. O algoritmo baseado em Arrefecimento Simulado replica as mudanças energéticas num sistema submetido a um processo de arrefecimento até convergir para um estado de equilíbrio sólido (El-Ghalazi, 2009).

O processo de arrefecimento do metal é realizado em duas etapas: Primeiro é submetido a uma temperatura alta no qual o metal se funde, na segunda etapa o arrefecimento é realizado lentamente até o metal solidificar. No caso do processo de arrefecimento se este não for controlado o metal obtido não atinge o equilíbrio térmico em cada temperatura, criando imperfeições no metal obtido (El- Ghazali, 2009).

A analogia com o algoritmo de Arrefecimento Simulado é feita na medida em que o algoritmo substitui a solução actual por uma próxima, dentro da vizinhança de soluções, escolhida considerando a função objectivo do problema e a função T, que representa a temperatura. A media que o algoritmo progride o valor de T é decrementado, assim o algoritmo converge para uma solução óptima local. Inicialmente qualquer movimento é aceita, o que permite explorar o espaço de soluções de forma exaustiva, mas a medida que o algoritmo progride apenas soluções que melhorem a função objectivo são aceites, neste processo a "temperatura" é diminuída (Madureira, 2009).

Não utiliza o conceito de memória, na medida em que não guarda informação recolhida durante o processo de pesquisa. É gerado uma vizinhança de forma aleatória em cada iteração, e movimenta-se de forma a escolher a solução que melhora a função objectivo. Se dentro da vizinhança gerada não existir nenhuma solução melhor então é seleccionada uma de acordo com uma probabilidade, que depende da temperatura actual e da degradação da função objectivo (El-Ghazali, 2009).

4.3.4.5.1. PARAMETRIZAÇÃO DO ALGORITMO DE ARREFECIMENTO SIMULADO

A semelhança do algoritmo de Pesquisa Tabu, no Arrefecimento Simulado tem de ser tomadas decisões que vão determinar a qualidade da solução produzida pelo algoritmo:

- **A escolha da probabilidade de aceitação de soluções piores p(n):** Na literatura é identificada a distribuição de Boltzman como forma de calcular a probabilidade (Madureira, 2009), (El-Ghazali, 2009).

$$P(n) = e^{-\frac{\Delta f_n}{T_n}} \quad (15)$$

Em que $\Delta f_n = f(x) - f(x_n)$ e T_n é a temperatura na iteração n.

- **Esquema de arrefecimento:** Define a temperatura do algoritmo em cada iteração, pelo que tem um papel importante no seu sucesso. Na bibliografia é possível encontrar vários esquemas de arrefecimento, no entanto o que apresenta uma maior robustez na solução é denominado por esquema geométrico (El-Ghazali, 2009).

$$T_{kl} = \alpha T_k \quad (16)$$

O factor α é um valor constante que varia de $0 < \alpha < 1$, Segundo a literatura é referido que o valor deve estar entre 0,55 e 0,99. *K* corresponde a iteração actual.

- **Número de iterações à mesma temperatura L:** Normalmente o valor L é definido em função da dimensão do problema. Para o algoritmo explorar convenientemente a vizinhança é aconselhável realizar um número de iterações à mesma temperatura. A definição do valor é constante e independente da temperatura (Madureira, 2009).

- **Definição da temperatura inicial:** A temperatura inicial não pode ser nem muito alta nem muito baixa, há que encontrar um equilíbrio entre os dois extremos. Por conveniência é assumido que a temperatura tem de ser alta o suficiente para que a probabilidade de aceitação de soluções inferiores seja de pelo menos 80%. Na bibliografia é referido que tem de se definir a probabilidade de aceitar "soluções piores" entre 0,7 e 0,8 calculando a temperatura inicial do algoritmo a partir deste pressuposto.

- **Escolha de um critério de paragem:** Tal como no algoritmo de Pesquisa Tabu, tem de escolher um critério de paragem para o algoritmo. Uma vez atingido este parâmetro o algoritmo termina.
    - Atingir determinada temperatura. É dos mais populares e normalmente o valor da temperatura é baixo;
    - Atingir uma quantidade determinada de iterações sem que a função objectivo seja melhorada;

De seguida será apresentado um algoritmo que exemplica o procedimento do Arrefecimento simulado. O algoritmo foi retirado de (Madureira, 2009).

4.3.4.5.2. ALGORITMO BASEADO NO ARREFECIMENTO SIMULADO

Algoritmo 3 - Algoritmo *de* Arrefecimento Simulado

```
Início
    Seleccionar a solução inicial x1 em X
        x1 = solução_inicial()
        f* = f(x1)
        x* = x1
        Tn = temperatura_inicial()
    Enquanto critério_paragem = falso fazer
        Para  k=1 até numero_iterações fazer
            x = gera_vizinho_aleatório()
            Se f(x) ≤ f(xn) então
                xn = x
                Se f(x) < f* então
                    x* = x
                    f* = f(x)
                FimSe
             Senão
                gerar aleatóriamente p ∈ [0,1]
                Se p ≤ p(n) então
                    xn+1 = x
                FimSe
            FimSe
        FimPara
        Tn+1 = αTn
   FimEnquanto
Fim
```

Figura 24 - Algoritmo baseado no arrefecimento simulado (Madureira, 2009)

O algoritmo segue o funcionamento explicado ao longo desta secção. É definida uma solução inicial da qual o algoritmo vai partir e a temperatura inicial. De seguida realiza uma série L de K iterações e gera um vizinho da solução actual de forma aleatória. É considerada também a possibilidade de não existir nenhuma solução melhor na vizinhança, pelo que tem de se aceitar uma pior. Para isso é definido a probabilidade *p(n)*. É gerado aleatoriamente um valor *p*, se esse valor for inferior à probabilidade de aceitação o valor é aceite, senão é rejeitado.

4.3.4.5.3. EXEMPLO ILUSTRATIVO

No exemplo retirado de (Madureira, 2009) é possível verificar que em cada iteração é gerada uma série de 5 iterações. A solução inicial do problema é gerada de forma aleatória e apresenta um valor para a função objectivo de f(x1)= 20. Os elementos da vizinhança são

obtidos a partir da troca de tarefas adjacentes. De acordo com o exemplo o valor de T_0 foi obtido a partir de uma probabilidade de aceitação p_0= 0,7. O esquema de arrefecimento escolhido é baseado na progressão geométrica e a taxa de arrefecimento $\alpha = 0{,}95$. O valor de Δf_n para a temperatura inicial foi calculado com base em 20% do tamanho da vizinhança, e tem como valor 17. A temperatura inicial foi obtida através de Lei de Boltzman é de $T_0 = -\frac{17}{\ln 0{,}7} = 47{,}66$.

T_n	L	Solução inicial x_n	$f(x_n)$	Solucão actual x	$f(x)$	p_n	p
47.66	1	5 3 2 4 1	20	5 2 3 4 1	14		
	2	5 2 3 4 1	14	5 2 3 1 4	17	0.6976[11]	0.1
	3	5 2 3 1 4	17	5 2 3 4 1	14		
	4	5 2 3 4 1	14	5 2 3 1 4	17	0.7046	0.9
	5	5 2 3 4 1	14	**5 2 4 3 1**	**11**		
45.277	1	5 2 4 3 1	11	2 5 4 3 1	11		
	2	2 5 4 3 1	11	5 2 4 3 1	11		
	3	5 2 4 3 1	11	5 2 3 4 1	14	0.6884[12]	0.4
	4	5 2 3 4 1	14	5 3 2 4 1	20	0.6869	0.3
	5	5 3 2 4 1	20	3 5 2 4 1	20		
43.013	1	3 5 2 4 1	20	5 3 2 4 1	20		
	2	5 3 2 4 1	20	3 5 2 4 1	20		
	3	3 5 2 4 1	20	3 5 4 2 1	32	0.6580	0.7
	4	3 5 2 4 1	20	3 2 5 4 1	14		
	5	3 2 5 4 1	14	3 5 2 4 1	20	0.6459	0.9
40.862	1	3 2 5 4 1	14	3 2 5 1 4	17	0.6437	0.9
	2	3 2 5 4 1	14	3 5 2 4 1	20	0.6437	0.2
	3	3 5 2 4 1	20	3 2 5 4 1	14		
	4	3 2 5 4 1	14	2 3 5 4 1	14		
	5	2 3 5 4 1	14	2 3 5 1 4	17	0.6421	0.4

Figura 25 - Exemplo ilustrativo (Madureira, 2009)

4.3.5. META-HEURÍSTICAS BASEADAS EM POPULAÇÕES

As Meta-Heurísticas baseadas em populações podem ser vistas como um processo de melhoramento numa população de soluções. Partem inicialmente de uma população de soluções e a seguir uma nova população é gerada, de seguida esta nova população é integrada na população actual recorrendo a mecanismos de selecção. O processo de pesquisa termina quando o critério de paragem é cumprido (Madureura, 2009), (El-Ghazali, 2009).

Na literatura é possível identificar que as Meta-Heurísticas baseadas nas populações são classificadas em Algoritmos evolucionários e em Inteligência de enxame. Alguns dos algoritmos pertencentes a essas categorias serão explicados mais a frente.

Os algoritmos classificados como evolucionários actuam sobre uma população de possíveis soluções representam uma classe de algoritmos que simulam a evolução das espécies,

baseados na evolução de uma população de indivíduos, onde só os mais fortes sobrevivem. Os algoritmos classificados como Inteligência dos enxames são tidos em conta como técnicas baseadas no comportamento colectivo de sistemas auto-organizados. São inspirados no comportamento de espécies como as formigas, as abelhas, as vespas, os peixes entre outros.

4.3.5.1. CONCEITOS COMUNS NAS META-HEURÍSTICAS BASEADAS EM POPULAÇÕES

A semelhança das Meta-Heurísticas baseadas na Pesquisa local, existem conceitos comuns aplicados às diferentes Meta-Heurísticas. Partem de uma população inicial de soluções e através de métodos iterativos geram uma nova população que substitui a população actual. Este processo é repetido enquanto o critério de paragem não for atingido.

- **População inicial:** A geração da população inicial tem um papel importante na eficiência e na efectividade do algoritmo. Um aspecto importante a considerar neste passo é a diversificação das populações, o que pode acarretar a convergência prematura do problema e não permitir uma exploração adequada do espaço de solução. Em (El-Ghazali, 2009) podem ser identificadas 4 estratégias utilizadas na geração da população inicial: **geração aleatória, a diversificação sequencial, a diversificação paralela e a inicialização heurística** (El-Ghazali, 2009)**.**
- **Critério de paragem:** Quando a população estagna na sua evolução, as MH baseadas em população tornam-se inutilizáveis. O critério de paragem lida com este facto. Normalmente o critério escolhido depende do problema em causa e do esforço computacional exigido. Os critérios de paragem utilizados nas Meta-Heurísticas baseadas em populações podem ser classificados da seguinte forma (El-Ghazali, 2009):
    - **Estáticos:** O critério de paragem é um valor pré-determinado e conhecido a *priori*. Limites dos recursos do CPU, número fixo de iterações ou um número máximo de avaliações da função objectivo são alguns dos utilizados neste contexto.
    - **Adaptivos:** O critério de paragem não é um valor fixo e não é conhecido. O número de iterações sem registar uma melhoria, quando

uma solução óptima ou satisfatória é encontrada, são alguns dos critérios utilizados neste contexto.

4.3.5.2. ALGORITMOS GENÉTICOS

A diferença das Meta-Heurísticas de Pesquisa Local, no âmbito do escalonamento as sequências são vistas como indivíduos que pertencem a uma população. Cada iteração do algoritmo é vista como uma geração nova. A população de uma nova geração é constituída pelos indivíduos que sobreviveram da última geração e os novos descendentes. O tamanho da nova população é constante em cada geração (Pinedo, 2009).

As gerações de soluções são geradas a partir da reprodução e mutação dos indivíduos que fizeram parte da geração anterior. No contexto do escalonamento uma mutação pode ser vista como a troca de tarefas adjacentes na próxima geração. Em cada uma das gerações apenas os indivíduos mais fortes sobrevivem, isto é, aqueles que apresentarem o melhor valor em termos de função objectivo (Pinedo, 2009).

Os Algoritmos Genéticos são constituídos por três etapas principais: A **selecção,** a **reprodução** e a **substituição.** Durante a selecção cria-se uma população provisória com alguns dos elementos da geração anterior. Apenas os indivíduos mais aptos tem mais probabilidade de transitar para a nova geração e sobreviver. Na segunda etapa são aplicados operadores reprodutivos de forma a gerar uma nova população. Finalmente, os indivíduos da população principal são substituídos pelos novos indivíduos (Madureira, 2009).

São, geralmente, utilizados nos problemas de optimização combinatória quando o espaço de procura é muito grande e os métodos mais tradicionais não conseguem responder de forma eficiente. A ligação entre um problema de optimização e os Algoritmos Genéticos são os cromossomas, nos que cada um representa uma solução num determinado espaço de soluções (Madureira, 2009), (Pinedo, 2009).

Em relação aos algoritmos baseados nas Meta-Heurísticas de Pesquisa Local, os Algoritmos Genéticos partem para a procura de uma solução com base num conjunto de soluções possíveis e não a partir de uma única solução. Para o seu correcto funcionamento é necessário manter uma diversificação suficiente na população para evitar a estagnação do algoritmo em máximos locais (Madureira, 2009).

Parte de uma população inicial de N indivíduos e em cada iteração o tamanho da população é mantido. Cada geração de soluções x_{n+1} é obtida a partir da evolução da geração de ordem *n*. Cada individuo representa uma solução, os melhores são seleccionados e cruzados entre si produzindo uma descendência de soluções. Cada solução da população actual é avaliada pelo seu enquadramento ou aptidão, que se pode traduzir na avaliação da função objectivo. A selecção dos melhores indivíduos em cada geração é realizada considerando o seu enquadramento. Os pares são submetidos à operação cruzamento com probabilidade T_c e cada descendente é submetido à operação de mutação com probabilidade T_m. Finalmente os piores indivíduos de cada geração são substituídos pelos descendentes, possivelmente mutados e o algoritmo continua a ser executado enquanto não for cumprido o critério de paragem.

Algoritmo 4: Algoritmo Genético

```
Inicio
    NGerações←0
    Geração de uma população inicial
    Definir M←N/2
    Enquanto (condição de paragem= falso) fazer
      Ngerações← Ngerações +1
      Para k←1 até M fazer
         Selecção de um par de  indivíduos
         Cruzamento
         Mutação
      FimPara
      Avaliação das soluções e selecção do melhor indivíduo
      Substituição dos "maus" indivíduos e criação da nova geração
    FimEnquanto
Fim
```

Figura 26 - Algoritmo Genético (Madureira, 2009)

4.3.5.3. *SCATTER SEARCH* ("PESQUISA DIFUSA")

A *scatter search* foi introduzida em 1977 por F. Glover e parte de uma população inicial denominada por Conjunto de referências. Utiliza combinações lineares de soluções com o objectivo de produzir soluções novas para as futuras gerações. Com vista a melhorar as soluções encontradas recorre a aplicação de heurísticas baseadas na Pesquisa Local.

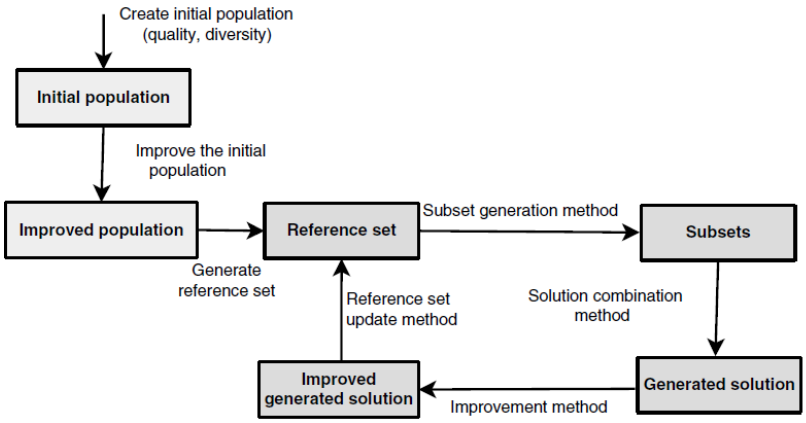


Figura 27 - Scatter Search (El-Ghazali, 2009)

Partindo de uma solução inicial são aplicadas Meta-Heurísticas baseadas na Pesquisa Local com vista a melhorar a solução inicial, onde são criadas as denominadas "soluções de teste". Depois é construído o conjunto de referências com base nas soluções de teste, onde são incluídas as melhores soluções. É gerado um sub-conjunto de soluções ao qual é aplicado métodos de combinação de soluções e depois métodos de melhoramento. O processo é repetido enquanto não for atingido um determinado critério de paragem.

4.3.5.4. INTELIGÊNCIA DO ENXAME

Esta classe de algoritmos é baseada no comportamento colectivo das formigas, abelhas, vespas, peixes, pássaros, entre outros. São desenvolvidos partindo do comportamento social destas espécies, que competem por alimentação. Dentro desta categoria destacam dois algoritmos particularmente: **Colonia de formigas** e **Optimização por nuvem de partículas** (El-Ghazali, 2009)**.**

A optimização por **Colonia de Formigas** foi introduzida por Marco Dorigo em 1992. A ideia fundamental destes algoritmos é imitar o comportamento real de uma formiga com o objectivo de resolver problemas de optimização. O interesse no comportamento das formigas está relacionado com o facto de as formigas realizarem tarefas complexas, como transportar comida e encontrar caminhos mais curtos, através do esforço colectivo da colonia. Durante a sua trajectória entre a fonte de comida e o formigueiro, as formigas depositam uma substância denominada por feromona, ao escolherem um caminho optam por aquele que possuir a maior concentração de feromonas, pois provavelmente será o mais curto. Assim, os algoritmos são desenvolvidos com base nesta lógica. A base reside na

parametrização probabilística que propõe o uso de feromona para definição de um caminho que conduza a solução óptima do problema. Informação mais detalhada sobre o funcionamento deste algoritmo e dos parâmetros que o constituem pode ser encontrada em (El-Ghazali, 2009), (Madureira, 2009).

Os algoritmos baseados na **Optimização por nuvens de partículas** tentam imitar o comportamento social de bandos de aves ou cardumes de peixes, que em movimentos coordenados e sincronizados tentam descobrir fontes de alimentos ou também como mecanismo de defesa. Foi proposto em 1995 por Eberhat e Kennedy (Kennedy, 1995). Em cada iteração do algoritmo as potenciais soluções, ou partículas, movimentam-se pelo espaço de pesquisa de acordo com uma regra que rege o comportamento. Esta regra depende de três factores (Madureira, 2009):

- **Inércia:** As partículas tendem a mover-se na direcção do movimento anterior;
- **Memória:** As partículas tendem a mover-se na direcção da melhor solução global;
- **Cooperação:** As partículas tendem a mover-se na direcção da melhor solução global encontrada até o momento.

Mais informação sobre este algoritmo pode ser encontrada em (El-Ghazali, 2009) e (Madureira, 2009).

4.4. RESUMO DO CAPÍTULO

Ao longo dos anos a preocupação por desenvolver métodos capazes de lidar com a complexidade dos problemas de optimização combinatória levou a que um esforço considerável fosse realizado na procura de soluções que permitam reduzir o esforço computacional investido na procura de soluções assim como a eficiência dos métodos que conduzem a essas soluções.

A escolha de um método para resolver um problema de escalonamento depende sempre das características particulares do problema, assim como dos recursos computacionais disponíveis e ainda há que considerar o factor económico. Na indústria o tempo é dinheiro. Pelo que existe uma preocupação crescente na procura de métodos que consigam produzir soluções óptimas ou satisfatórias para os problemas em tempo útil.

A dimensão do problema condiciona a escolha do método de resolução de um problema. Quando a dimensão tende a aumentar os métodos baseados nas Meta-Heurísticas são

eficientes na procura de uma solução em tempo útil, no entanto ainda não é claro para que tipo de problema cada uma das Meta-Heurísticas é mais adequada. Cada problema é um caso de estudo a parte, pelo que do leque de soluções disponíveis é escolhido aquele que se adequar mais às características do problema, podendo ainda desenvolver vários métodos de resolução e comparar os resultados obtidos através da sua aplicação.

5. PROTÓTIPO DESENVOLVIDO E TESTES COMPUTACIONAIS

5.1. FERRAMENTA DESENVOLVIDA

No âmbito deste trabalho foi desenvolvida uma ferramenta que aplica uma Meta-heurística baseada em Pesquisa Local aos problemas de Máquina Única. As instâncias testadas no algoritmo são do tipo soma pesada dos atrasos pesados. A ferramenta foi desenvolvida recorrendo ao Visual Studio 2013 e a linguagem de programação utilizada foi o C#.

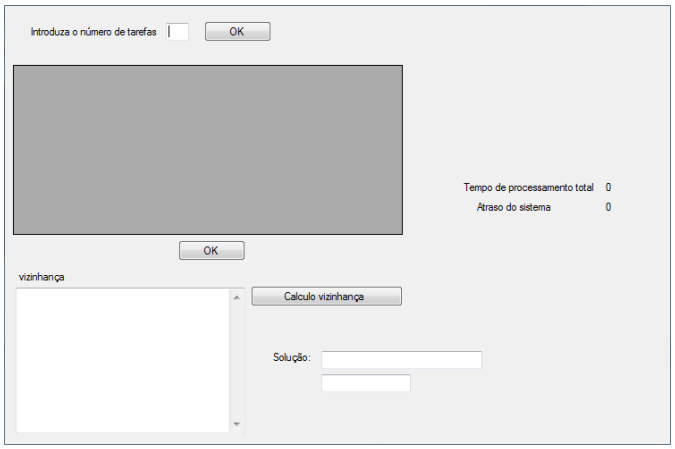


Figura 28 - Protótipo desenvolvido

O *layout* do protótipo é o apresentado na Figura 28. Ao iniciar o programa o utilizador insere o número de máquinas que serão alvo do escalonamento e a seguir é apresentada uma tabela, onde são inseridos os dados. As filas dependem do número de máquinas inseridas, no entanto o número de colunas é fixo independentemente do número de máquinas inserido. Os dados solicitados são: **Número de tarefa, tempo de processamento, data de entrega e penalização.** Para o algoritmo escolhido foram assumidos alguns pressupostos:

- Não existem tempos de *set-up*;
- Todas as tarefas estão disponíveis no instante *t=0*, não existem datas de lançamento;
- Tempos de processamento e data de entrega são previamente conhecidos.

No canto inferior esquerdo é apresentada a composição da vizinhança do problema. Neste contexto foi escolhido o método de troca de tarefas adjacentes como mecanismo de geração da vizinhança.

É possível organizar os dados da tabela por data de lançamento ou tempo de processamento. Para isso basta realizar um clique no cabeçalho correspondente, depois de inseridos os dados, para obter a ordem desejada. A disposição dos dados na tabela determina a solução inicial da qual o algoritmo parte para a resolução do problema. Os resultados são apresentados através da sequência assim como o atraso pesado respectivo.

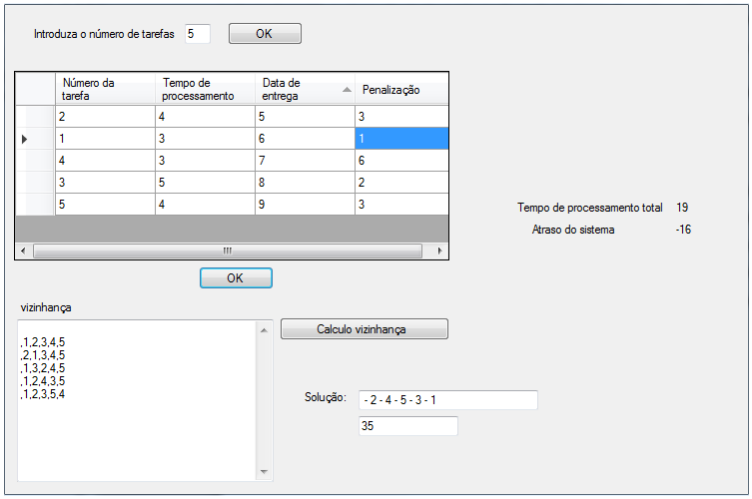


Figura 29 - Exemplo ilustrativo

Na Figura 29 foi simulado um ambiente constituído por 5 máquinas. Os dados foram organizados por data de lançamento, através da regra de prioridade EDD, e o resultado obtido pelo algoritmo foi a sequência [2 4 5 3 1] com um atraso pesado associado de 35.

5.2. PLANO DE TESTE

Para o estudo computacional foram consideradas 5 instâncias do problema de sequenciamento de Máquina Única cuja função objectivo é a de minimizar o atraso pesado, "*Weighted Tardiness"*. As instâncias foram retiradas da *OR-Library* [URL 2].

Foram avaliados os resultados obtidos pelo algoritmo baseado na Pesquisa Local desenvolvido no contexto deste trabalho, a regra de prioridade EDD e os algoritmos "*Shifting Bottelneck"* e "*Shifting Bottleneck / sum wT"*. Estes dois últimos foram aplicados recorrendo ao Software de escalonamento Lekin.

5.3. RESULTADOS COMPUTACIONAIS

Na Tabela 5 É possível verificar o desempenho do algoritmo partindo de soluções baseadas nas regras de prioridade SPT, LPT e EDD.

Tabela 5 – Resultados obtidos pelo algoritmo

	PL+EDD	PL+SPT	PL+LPT
WTA	913	6010	11313
WTB	2574	7445	7445
WTC	1515	3472	8576
WTD	2531	4716	4976
WTE	3532	2280	11575

Na Figura 30 é apresentado um gráfico comparativo do desempenho do algoritmo partindo das diferentes soluções iniciais, geradas pelas regras de prioridade EDD, SPT e LPT. É possível verificar que as soluções encontradas pelo algoritmo quando se parte de uma solução gerada pela regra EDD são melhores quando comparadas com as soluções encontradas pelo algoritmo partindo das outras regras de prioridade. A excepção ocorre na instância WTE, onde o algoritmo encontrou uma melhor solução partindo da regra SPT. Os piores resultados, em termos de função-objectivo, foram apresentados partindo da regra LPT.

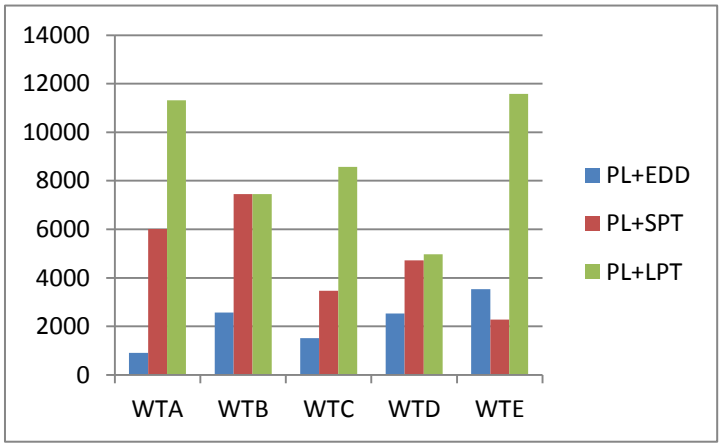


Figura 30 - Resultados do Algoritmo baseado na PL

Na Tabela 6 é possível observar os resultados obtidos através da aplicação do Algoritmo de Pesquisa Local, a regra de prioridade EDD e os algoritmos baseado no SB. No algoritmo de Pesquisa Local os dados foram organizados por data de lançamento, de acordo com a regra de prioridade EDD.

Tabela 6 - Resultados obtidos

	Óptimo	EDD	PL	SB	SB/ WT
WT40A	913	1588	913	913	965
WT40B	1225	5226	2574	1502	1889
WT40C	537	3051	1515	573	1296
WT40D	2094	5527	2531	2571	3190
WT40E	990	4030	3532	1296	1641

Na Figura 31 é possível observar um gráfico comparativo entre os métodos de optimização aplicados em relação ao valor óptimo encontrado até a data para as instâncias. Do gráfico é possível concluir que o algoritmo baseado no *Shifting Bottelneck* conseguiu os melhores resultados dentro do contexto no qual foram implementados os algoritmos. A regra de prioridade EDD foi aquela que apresentou piores resultados de escalonamento.

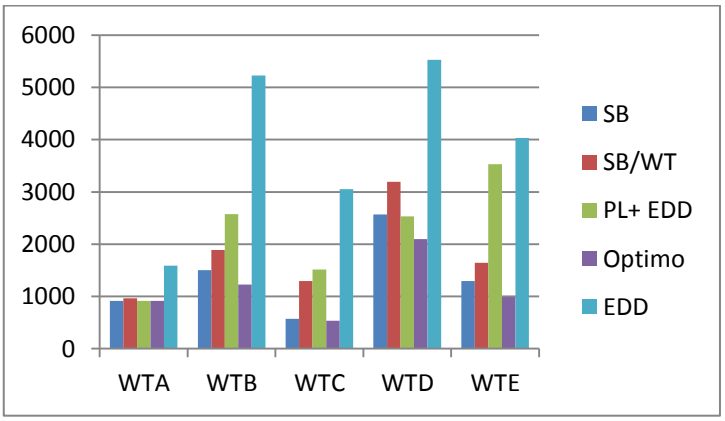


Figura 31 - Resultados dos algoritmos na resolução de instâncias WT

5.4. DISCUSSÃO DE RESULTADOS

Os testes realizados basearam-se na resolução de uma instância específica dos problemas de escalonamento, os problemas de atraso pesado. É possível analisar, com base nos resultados, a qualidade das respostas obtidas.

Verificou-se uma qualidade superior, em termos de função objectivo, dos resultados obtidos pelo algoritmo partindo de uma solução inicial baseada na regra de prioridade EDD, a excepção da instância WT40E, onde o algoritmo partiu de uma solução baseada na

regra SPT. Quando partiu de uma solução baseada na regra LPT o algoritmo apresentou os piores resultados.

Para a primeira instância analisada, WT40A, o algoritmo baseado na PL assim como o algoritmo *SB* conseguiram chegar a melhor solução encontrada até a data. A regra de prioridade EDD apresenta o pior resultado entre os métodos de resolução aplicados. Para a segunda instância os algoritmos baseados no *SB* apresentaram os melhores resultados, no entanto nenhum chegou à melhor solução encontrada até a data. Esta situação repete-se para as restantes instâncias do problema. Os algoritmos baseados no SB apresentaram um valor para a função objectivo menor do que o algoritmo baseado na Pesquisa Local e a regra de prioridade EDD.

Tabela 7 - Desvio dos resultados

	EDD	PL	SB	SB/WT
WTA	42,5	0,0	0,0	5,4
WTB	76,6	52,4	18,4	35,2
WTC	82,4	64,6	6,3	58,6
WTD	62,1	17,3	18,6	34,4
WTE	75,4	72,0	23,6	39,7
Média	**67,8**	**41,2**	**13,4**	**34,6**

Na Tabela 7 é possível observar o cálculo do desvio do resultado encontrado por cada método em cada uma das instâncias. O desvio é calculado em base ao melhor resultado encontrado até a data. Através dos valores médios é possível concluir que os algoritmos baseados no SB apresentam melhores resultados, face ao algoritmo baseado na Pesquisa Local e a regra de prioridade EDD. Isto está relacionado com o facto do algoritmo de Pesquisa Local ficar limitado aos mínimos locais, não conseguindo explorar o espaço de solução de cada instância na sua totalidade. Além da vizinhança de cada solução parcial estar limitada à troca adjacente de tarefas, isto significa que dentro dessa vizinhança poderá existir melhores soluções se o método fosse de geração fosse outro dos mencionados na secção 4.

A solução eficiente de um problema de escalonamento requer a aplicação de diferentes métodos de optimização. Não existe um método "geral" para aplicar num determinado problema. A parametrização dos algoritmos é importante, já que vão definir a qualidade da resposta dos problemas estudados. A qualidade de uma resposta também depende da

robustez dos algoritmos que forem aplicados, assim como dos recursos disponíveis para a sua procura.

5.5. RESUMO DO CAPÍTULO

No capítulo 5 foi apresentada a ferramenta de apoio desenvolvida no âmbito desta dissertação assim como os testes computacionais realizados. O desempenho do algoritmo foi verificado ao executar o algoritmo partindo de 3 soluções iniciais, baseadas nas regras de prioridade EDD, SPT e LPT. Os resultados permitem verificar que foram obtidos melhores resultados partindo da regra EDD.

A qualidade das respostas produzidas pelo algoritmo, foi analisada com base nas respostas obtidas aplicando outros métodos de aproximação, a regra de prioridade EDD e dois algoritmos baseados no SB. Os resultados permitiram verificar que os algoritmos baseados no SB apresentam soluções mais perto do valor óptimo, se comparadas com aquelas obtidas pelo algoritmo baseado na PL.

Foi possível verificar que os métodos de aproximação conseguem produzir respostas com qualidade de forma eficiente.

6. CONCLUSÃO

Ao longo deste texto foram sendo apresentadas conclusões que permitiram sustentar as opções de desenvolvimento efectuadas ao longo do trabalho. O problema do escalonamento é um caso que tem sido exaustivamente estudado ao longo do tempo. O seu interesse reside no facto de que um esquema de escalonamento optimizado se traduz num processo de produção eficiente, no qual é retirado maior desempenho aos componentes que fazem parte do sistema de produção.

Os problemas de escalonamento estudados em ambientes académicos distam muito da realidade, já que em ambientes industriais reais existem situações que se podem manifestar, comprometendo a planificação realizada e que em ambientes académico não se consideram.

Os problemas de escalonamento são classificados como sendo NP-difíceis, isto significa que a dificuldade cresce de forma exponencial com a dimensão do problema. Para a resolução dos problemas existem muitos algoritmos capazes de resolver as mais complexas das situações, no entanto uma conclusão importante que se pode retirar deste trabalho é que não existe um algoritmo específico para a resolução de um problema de escalonamento específico. Existem métodos de optimização exactos, no entanto a sua utilização está condicionada à dimensão do problema, a medida que esta aumenta o algoritmo torna-se

ineficiente a nível de tempo. O tempo que se investe na procura de uma solução é importante, pelo que existe uma constante preocupação em desenvolver metodologias capazes de produzir resultados satisfatórios em tempo útil.

As Meta-Heurísticas conseguem cumprir o objectivo de produzir soluções de qualidade em tempo útil. Neste trabalho foi dado ênfase especial às Meta-Heurísticas baseadas na Pesquisa Local. A base do seu funcionamento está no mecanismo de geração das vizinhanças. A vizinhança da solução principal define o espaço de solução do problema, pelo que o desempenho dos algoritmos é dependente da robustez deste mecanismo. Através da pesquisa elaborada no âmbito deste trabalho foram identificados vários métodos de geração baseados na permutação de tarefas, no caso particular deste trabalho o mecanismo escolhido foi o de trocar tarefas adjacentes.

Os resultados obtidos através do algoritmo implementado, quando comparados com os implementados com o Lekin, apresentam um desvio maior em relação à melhor solução encontrada para as instâncias testadas. Esta situação acontece devido à natureza dos algoritmos do Lekin, baseados no *Shifting Bottelneck*. A exploração do espaço de solução foi realizada de uma forma mais exaustiva do que no algoritmo desenvolvido, baseado na Pesquisa Local. Através do mecanismo escolhido para a geração da vizinhança, a vizinhança fica reduzida a N-1 elementos, em que N corresponde a dimensão do problema. Além de existir a limitação conhecida de ficar preso nos mínimos locais.

Esta situação torna evidente que para obter um plano de escalonamento óptimo é necessário implementar diferentes métodos até encontrar o que melhor se adaptar às necessidades do sistema. A parametrização dos algoritmos assume uma importância relevante neste aspecto devido aos resultados que podem apresentar se a parametrização for realizada de forma deficiente. Existem regras e sugestões na parametrização dos algoritmos aqui apresentados que devem ser seguidas, para uma correcta utilização dos métodos de optimização.

Como perspectiva de desenvolvimento futuro sugere-se a aplicação de outros métodos de optimização, aplicando algoritmos baseado na Pesquisa Tabu, Arrefecimento Simulado ou métodos baseados nas Populações, que são alguns dos métodos identificados na bibliografia como testados em ambientes de escalonamento e tendo produzido resultados satisfatórios.

Referências Bibliográficas

[1] Andersen, Michael; Bräsel Heidemarie; Engelhardt Frank; Werner, Frank – A Library of Scheduling Algorithm, 2010

[2] Baker, Keneth; Trietsch, Dan – Principles of Sequencing and Scheduling, New Jersey 2009.

[3] Blazewicz, Jacek; Ecker, Klaus; Pesch, Erwin; Smith, Günter; Shaw, Michael – Handbook of Scheduling, New York 2007.

[4] Brucker, Peter – Scheduling Algorithms, New York 2007

[5] Cavaco, Ismale; Ávila, Paulo – Tipologia dos Sistemas de Produção, 2008

[6] Costa Lemos, Natália Maria - Comparação de Escalonamento num Ambiente de Linha de Produção Hibrida com um Ambiente de Linhas de Produção Paralelas Um Caso Industrial, 2014.

[7] El-Ghazali, Talibi- Metaheuristics from Design to Implementation, New Jersey 2009.

[8] Kennedy, James; Eberhar, Russel - Particle Swarm Optimization, 1995.

[9] Madureira, Ana – Aplicação de Meta-heurísticas ao Problema de Escalonamento em Ambiente Dinâmico de Produção discreta, 2003.

[10] Madureira, Ana - Técnicas Emergentes de Optimização no Suporte à Tomada de Decisão, Porto 2009.

[11] Madureira, Ana; Santos, Joaquin; Pereira Ivo - MASDScheGATS - Scheduling System for Dynamic Manufacturing Environmemts, 2009.

[12] Madureira, Ana; Pereira, Ivo; Sousa, Nelson; Ávila, Paulo; Bastos, João – Scheduling a Cutting and Treatment Stainless Steel Sheet Line with Self – Management Capabilities, 2011.

[13] Madureira, A; Gomes, S; Cunha, B; Pereira, J.P; Santos, J.M.; Pereira, I - Prototype of an Adaptive Decision Support System for Interactive Scheduling with MetaCognition and User Modeling Experience, 2014.

[14] Papadimitrou, Christos H.; Steiglitz Kenneth – Combinatorial Optimization: Algortithms and Complexity, Toronto 1998.

[15] Papadimitrou, Christos H – Computational Complexity, USA 1994

[16] Pinedo, Michael L – Planning and Scheduling in Manfacturing and Services, New York 2009.

[17] Pinedo, Michael L- Scheduling Theory, Algorithms, and Systems 4th edition, New York 2012.

[18] Pirlot, Marc – General Local search methods, 1996.

[19] Rothlauf, Franz – Design of Modern Heuristics, 2011

[20] Schrijver Alexander – A Course in Combinatorial Optimization, Neteherlands, 2013

[21] Siarry, Patrick; Michalewicz Zbigniew – Advances in Meteheuristics for Hard Optimization, Berlin 2008.

URL

[1] http://www.math.ovgu.de/Lisa/Papers.html

[2] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html

[3] http://neos-guide.org/

[4] http://www.optirisk-systems.com/products_fortmp.asp

[5] http://community.stern.nyu.edu/om/software/lekin/index.htm

[6] http://www.optirisk-systems.com/products_fortmp.asp

Anexo A.

As instâncias do problema “Weighted Tardiness” com 40 tarefas foram retiradas Or-Library.

Tabela 8 - Instância do problema WTA

WTA			
i	p_i	d_i	w_i
1	26	1588	1
2	24	1620	10
3	79	1731	9
4	46	1773	10
5	32	1694	10
6	35	1487	4
7	73	1566	3
8	74	1844	2
9	14	1727	10
10	67	1636	3
11	86	1599	7
12	46	1539	3
13	78	1855	1
14	40	1645	3
15	29	1709	10
16	94	1660	4
17	64	1582	7
18	27	1836	7
19	90	1484	4
20	55	1559	7
21	35	1772	5
22	52	1510	3
23	36	1512	5
24	69	1795	4
25	85	1522	9
26	95	1509	5
27	14	1598	2
28	78	1658	8
29	37	1826	10
30	86	1628	4
31	44	1650	7
32	28	1833	4
33	39	1627	9
34	12	1528	5
35	30	1541	7
36	68	1497	7
37	70	1481	5
38	9	1446	10
39	49	1579	1
40	50	1814	3

Anexo B.

Tabela 9 - Instância do problema WTB

WTB			
i	p_i	d_i	w_i
1	56	1687	1
2	25	1738	9
3	76	1663	9
4	35	1480	9
5	28	1504	5
6	52	1826	1
7	21	1722	4
8	32	1660	3
9	64	1594	2
10	67	1445	8
11	48	1704	2
12	100	1660	3
13	94	1715	7
14	87	1701	5
15	39	1679	3
16	18	1516	5
17	78	1658	2
18	80	1611	2
19	56	1502	2
20	72	1685	4
21	4	1614	9
22	70	1647	9
23	36	1689	6
24	46	1615	8
25	85	1524	9
26	31	1800	7
27	96	1654	5
28	30	1752	2
29	66	1456	1
30	92	1452	6
31	33	1801	4
32	18	1713	6
33	19	1761	1
34	34	1513	2
35	18	1759	10
36	4	1484	10
37	42	1821	6
38	94	1448	5
39	4	16666	4
40	89	1611	3

Anexo C.

Tabela 10 - Instância do problema WTC

WTC			
i	p_i	d_i	w_i
1	1	1452	10
2	49	1565	4
3	35	1588	7
4	83	1319	3
5	75	1436	6
6	64	1434	7
7	20	1573	5
8	84	1427	10
9	31	1593	5
10	88	1432	10
11	27	1428	2
12	88	1549	1
13	21	1565	7
14	32	1312	7
15	12	1614	2
16	20	1362	8
17	26	1643	3
18	64	1536	8
19	6	1372	8
20	11	1490	8
21	54	1631	10
22	2	1338	1
23	21	1336	1
24	94	1487	3
25	44	1361	6
26	19	1363	7
27	45	1583	2
28	6	1652	4
29	61	1396	6
30	41	1376	5
31	45	1319	7
32	86	1369	4
33	98	1341	4
34	45	1434	9
35	66	1319	5
36	77	1296	6
37	76	1644	1
38	64	1418	9
39	31	1421	9
40	25	1338	4

Anexo D.

Tabela 11 - Instância do problema WTD

WTD			
i	p_i	d_i	w_i
1	71	1419	6
2	58	1682	3
3	89	1683	9
4	62	1617	3
5	8	1703	8
6	31	1549	8
7	74	141	3
8	52	1634	8
9	71	1580	10
10	85	1588	4
11	1	1694	7
12	77	1574	8
13	35	1548	10
14	30	1730	5
15	96	1535	5
16	12	1438	6
17	4	1501	5
18	29	1504	10
19	64	1587	3
20	34	1687	6
21	8	1472	9
22	98	1507	4
23	86	1389	3
24	22	1454	2
25	6	1404	7
26	6	1522	10
27	24	1526	6
28	61	1681	10
29	86	1506	9
30	76	1584	3
31	17	1720	2
32	36	1767	6
33	63	1621	6
34	83	1677	9
35	81	1487	10
36	37	1513	9
37	80	1592	7
38	56	1620	5
39	11	1771	5
40	57	1712	6

Anexo E.

Tabela 12 - Instância do problema WTE

WTE			
i	p_i	d_i	w_i
1	7	1471	5
2	70	1794	10
3	52	1514	1
4	52	1473	2
5	86	1702	2
6	66	1583	8
7	70	1640	8
8	60	1683	5
9	65	1491	6
10	70	1519	4
11	27	1702	8
12	41	1527	4
13	42	1701	1
14	88	1675	1
15	21	1421	6
16	15	1728	5
17	40	1639	10
18	80	1742	7
19	28	1749	2
20	7	1653	7
21	13	1441	5
22	58	1722	8
23	4	1691	7
24	43	1612	7
25	41	1424	6
26	89	1615	2
27	80	1809	6
28	54	1533	6
29	34	1648	1
30	92	1702	8
31	66	1450	8
32	72	1614	5
33	29	1675	7
34	40	1435	6
35	53	1441	6
36	35	1485	4
37	91	1746	8
38	58	1732	6
39	6	1727	5
40	82	1612	7