• U C •

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

DEPARTAMENTO DE
ENGENHARIA MECÂNICA

Utilização do *Simulated Annealing* na resolução de problemas no planeamento de produção

Dissertação apresentada para a obtenção do grau de Mestre em Engenharia e Gestão Industrial

Autor

Nuno Filipe Pedrosa Loureiro

Orientador
Professor Doutor Cristóvão Silva

Júri

Presidente	**Professor Doutor** José Luís Ferreira Afonso **Professor da Universidade de Coimbra**
Vogais	**Professor Doutor** Luís Miguel Domingues Fernandes Ferreira **Professor da Universidade de Aveiro**
	Professor Doutor Cristóvão Silva **Professor da Universidade de Coimbra**

Coimbra, Julho, 2014

“Nada é particularmente difícil se for dividido em pequenas tarefas.”

Henry Ford

Agradecimentos

O trabalho que aqui se apresenta só foi possível graças à colaboração e apoio de algumas pessoas, às quais não posso deixar de prestar o meu reconhecimento, nomeadamente, à minha família, pai, mãe e irmãs, aos amigos e a alguns colegas. Agradeço ainda ao meu orientador Cristóvão Silva pela disponibilidade e colaboração indispensáveis à execução deste trabalho.

Resumo

O objetivo deste trabalho é testar a utilização da meta-heurística *Simulated Annealing* na resolução de problemas comuns ao planeamento da produção. São desenvolvidos algoritmos de *Simulated Annealing* adaptados aos problemas concretos em estudo, o sequenciamento em máquina única e a afetação de tarefas em máquinas paralelas, de forma a obter resultados que possam ser utilizados para uma análise comparativa com valores de referência já conhecidos para estes problemas. Por fim são tiradas conclusões relativas ao desempenho deste método na obtenção de soluções que possam ser consideradas "boas".

Palavras-chave: Sequenciamento, Otimização, *Simulated Annealing*, Afetação de tarefas, Meta-Heurísticas, Planeamento Produção.

Abstract

The main objective of this work is to test the use of meta-heuristic Simulated Annealing in solving common production planning problems. Simulated Annealing algorithms adapted to the specific problems under study, the single machine scheduling and the allocation of tasks to parallel machines, are developed in order to obtain results which can be used for a comparative analysis with reference values known for these problems.

Finally are taken conclusions about the performance of this method in obtaining solutions that can be considered "good".

Keywords Scheduling, Optimization, *Simulated Annealing*, Allocation of tasks, Meta-Heuristics, Production Planning.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIMBOLOGIA

T– Tardiness

WT – Weighted Tardiness

M - *Makespan*

1. INTRODUÇÃO

O trabalho descrito nesta tese tem como objetivo utilizar a meta-heurística *Simulated Annealing* para resolver problemas de sequenciamento bastante comuns no planeamento da produção, e desta forma perceber até que ponto é que este tipo de modelo pode ser útil para resolver este tipo de problemas. Um dos problemas a resolver tem como base o sequenciamento de tarefas a serem executadas numa única máquina ao passo que o outro passa pela afetação de tarefas a duas máquinas distintas que funcionam em paralelo.

Os métodos tradicionais de otimização exata caracterizam-se pela rigidez dos seus modelos matemáticos representados através dos seus teoremas, dificultando a representação de situações reais cada vez mais complexas e dinâmicas. Este problema, de falta de flexibilidade começou a ficar resolvido no momento em que se passaram a associar técnicas de otimização com as ferramentas de busca heurística. Assim e dada a complexidade dos problemas em causa, ir-se-ão aplicar métodos que visam a obtenção eficiente de soluções satisfatórias em tempo aceitável, mas que não são necessariamente as soluções ótimas. Com este fim, utilizar-se-ão, para resolver os problemas em estudo nesta tese, métodos heurísticos baseados em pesquisa local, mais propriamente meta-heurísticas.

De forma a testar a viabilidade destas ferramentas heurísticas na resolução de problemas de sequenciamento, desenvolveram-se algoritmos adaptados a estes problemas em específico e implementaram-se estes mesmos algoritmos com recurso a ferramentas informáticas que por fim permitiram efetuar um estudo computacional que permitiu avaliar os resultados da utilização da meta-heurística *Simulated Annealing* na resolução destes problemas, comparando-os com valores de referência conhecidos.

Posto isto, esta tese é composta por 6 capítulos.

No capítulo 2 é feita uma introdução geral aos conceitos teóricos abordados nesta tese, como o sequenciamento em geral e os problemas relacionados com o sequenciamento em particular enfatizando a descrição mais detalhada dos problemas em máquina única e em máquinas paralelas, por serem os problemas em estudo neste trabalho. Em seguida é feita uma introdução dos conceitos relacionados com investigação operacional, modelação

matemática, meta-heurísticas e aprofundada a componente teórica em que se baseia a aplicação do *Simulated Annealing*.

No capítulo 3 é feita uma descrição dos dois tipos de problemas em estudo, sequenciamento em máquina única e afetação de tarefas em máquinas paralelas.

No capítulo 4 são descritos os modelos e as equações relacionadas com as medidas de desempenho estudadas neste trabalho.

No capítulo 5 é explicada de forma detalhada, a implementação computacional dos algoritmos, demonstrando excertos do código implementado em Visual Basic for Applications, acompanhado de uma explicação da sua funcionalidade específica para uma aplicação correta do algoritmo.

No capítulo 6 são apresentados e discutidos os resultados obtidos aquando da aplicação dos algoritmos desenvolvidos aos problemas concretos em estudo.

No capítulo 7 são explicitadas as principais conclusões tiradas deste trabalho e sugeridos alguns potenciais desenvolvimentos futuros.

2. REVISÃO DE CONCEITOS

2.1. Sequenciamento ("Scheduling")

Os sistemas produtivos envolvem uma grande quantidade de decisões que devem ser tomadas ao nível do planeamento de forma a cumprir os prazos de entrega dos produtos mas também a minimizar os custos o mais possível, sendo que esta minimização de custos pode advir do aumento da eficácia da relação matéria-prima – produto acabado, da otimização dos recursos necessários à produção, sejam eles ferramentas ou recursos humanos ou da minimização dos tempos totais de processamento.

Assim, no atual ambiente competitivo, o sequenciamento e planeamento tornaram-se uma necessidade para sobrevivência no mercado. As empresas devem esforçar-se ao máximo para cumprir as datas agendadas com os seus clientes. O fracasso deste compromisso pode resultar numa perda significativa de imagem da empresa perante os clientes. (Pinedo, 2002)

A maioria das aplicações de sequenciamento descritas na literatura foram motivadas por problemas de produção. No entanto a sua aplicação pode ser considerada em diversas outras áreas, são exemplos de problemas de sequenciamento: o atendimento de doentes num hospital ou de clientes num supermercado ou restaurante, a reparação de automóveis numa oficina, ou o controlo de tráfego num aeroporto.

O planeamento da produção envolve frequentemente a resolução de problemas de sequenciamento ("scheduling") com um impacto significativo no desempenho das organizações. O sequenciamento lida com a atribuição de recursos escassos a tarefas ao longo do tempo. Trata-se de um processo de tomada de decisão com o objetivo de otimizar um ou mais objetivos (Pinedo, 2002).

Dado o facto de tratarmos nesta tese de problemas relacionados com a gestão da produção assumir-se-á o sequenciamento como a afetação ótima no tempo de recursos escassos (máquinas) a tarefas, sujeitas a restrições, tais como o facto de nenhuma máquina processar mais que uma tarefa no mesmo instante e nenhuma tarefa ser processada em mais que uma máquina.

2.1.1. Problemas de Sequenciamento

Os problemas de sequenciamento (*scheduling*) têm, de uma forma geral, como objetivo a definição dos instantes temporais em que um conjunto de tarefas são processadas por um conjunto de recursos de forma a minimizar uma medida tipicamente relacionada com o tempo. Este tipo de problemas tem sido estudado desde há largas décadas (Brucker, 2004). Uma das áreas onde este tipo de problemas tem mais aplicação é no planeamento da produção.

Devido à complexidade dos problemas de sequenciamento da produção e à escassez de tempo de que geralmente se dispõe para a sua resolução, na grande maioria dos casos, o objetivo não passa por determinar a solução ótima para o problemas mas antes uma solução melhor que a atual, recorrendo ao que geralmente se designam por métodos de aproximação.

2.1.1.1. Abordagem aos problemas

Os problemas de sequenciamento têm sido tradicionalmente abordados como problemas de otimização sujeitos a restrições, cujos elementos básicos são as máquinas e as tarefas. Segundo Baker (1974) a abordagem aos problemas de sequenciamento passa por quatro fazes principais: a formulação, a análise, a síntese e a avaliação.

Na formulação é identificado o tipo de problema e determinado o critério que orientará o processo de decisão.

A análise consiste no processo de examinar pormenorizadamente os elementos do problema e as relações entre eles. Nesta fase são determinadas as variáveis de decisão, a relação entre elas e as restrições a obedecer.

Na fase de síntese são constituídas soluções admissíveis alternativas, caracterizando as diferentes linhas de ação disponíveis.

A avaliação é o processo de comparação dessas alternativas e da escolha da melhor solução para o problema, com base nos critérios desenvolvidos na fase de formulação do problema.

2.1.1.2. Diagrama de Gantt

Para representar graficamente as tarefas a serem realizadas num dado recurso (máquina) ao longo do tempo normalmente é utilizado o Diagrama de Gantt, também chamado de diagrama de barras. Este diagrama tem no eixo das ordenadas o recurso utilizado

e no eixo das abscissas uma linha de tempo, e apresenta depois uma barra. As divisões desta barra representam os instantes dos principais eventos da programação da produção. Os eventos mais importantes representados pelos diagrama de Gantt são o início e o fim do processamento de cada tarefa, como podemos observar no exemplo representado na figura 2.1.

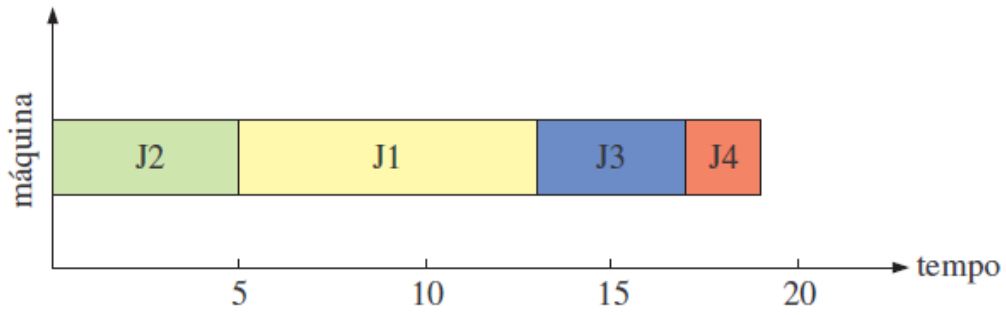

Figura 2.1 - Diagrama de Gantt (Júnior, 2007)

Num diagrama deste género, cada tarefa é representada por uma cor, e a alocação dessa tarefa a uma máquina é representada por uma barra da cor correspondente. A barra prolonga-se durante os períodos de tempo nos quais a máquina estará ocupada a realizar essa tarefa.

Na Figura 2.1 a tarefa (job) 2 é a primeira a ser executada e começa a ser processada no instante 0. Segue-se a tarefa 1 que começa a ser executada no instante 5 e termina no instante 13. Com o Diagrama de Gantt é possível perceber, de forma clara, como irão ser distribuídas as operações das tarefas pelas respetivas máquinas ao longo do tempo.

2.1.1.3. Classificação de problemas de sequenciamento

Os problemas de sequenciamento podem ser classificados pela notação $\alpha|\beta|\gamma$, que foi proposta por Lawler (1983), em que:

- α - especifica o tipo de problema (máquina única, máquinas paralelas, etc);
- β - representa as características das tarefas, se o campo β estiver vazio assume-se que todas as tarefas estão disponíveis para processamento ao mesmo tempo, não é permitida interrupção na sua execução e não existem restrições de precedência;
- γ - define os critérios de otimização ou as medidas de desempenho, na resolução do problema esta característica toma a forma de função objetivo, como exemplos de

funções objetivo temos a minimização do número de tarefas em atraso ou a minimização do tempo total de execução de todas as tarefas.

2.1.1.4. Tipos de Problemas de sequenciamento

Nesta tese abordaremos dois tipos de problema de sequenciamento em específico, o sequenciamento em máquina única e a afetação de tarefas em máquinas paralelas

2.1.1.4.1. Problemas Máquina única

Nos problemas de máquina única considera-se uma única máquina em funcionamento que irá realizar um conjunto de tarefas. É um problema elementar de sequenciamento em que o objetivo passa por organizar a sequência das tarefas a serem executadas de forma a minimizar um determinado parâmetro, no caso deste trabalho o objetivo é minimizar atraso total ou o atraso pesado total. Assume-se, em geral, que todas as tarefas estão disponíveis no instante de tempo t=0 e que os tempos de processamento são conhecidos e independentes da sequência determinada.

Segundo Pinedo (2008),os resultados que podem ser obtidos para os problemas do sequenciamento em máquina única não só fornecem o conhecimento para o ambiente neste tipo de problemas, como também fornecem base para heurísticas aplicáveis a ambientes mais complexos. Na prática, os problemas de programação em ambientes mais complicados são frequentemente decompostos em sub-problemas de máquina única.

A Figura 2.2 representa o esquema de funcionamento da execução de tarefas (job) em máquina única.

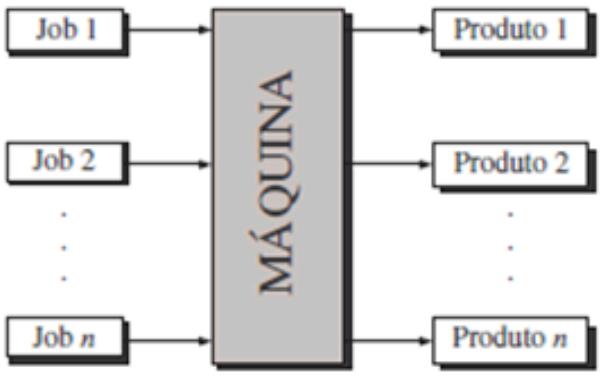

Figura 2.2 - Esquema do processo de execução de tarefas em máquina única

2.1.1.4.2. Problemas em Máquinas paralelas

Nos problemas em máquinas paralelas considera-se um conjunto de máquinas a funcionar em paralelo e um conjunto de tarefas com datas de disponibilidade e de entrega.

Considera-se ainda que existem tempos de preparação que dependem da sequência das tarefas e que os tempos de processamento das tarefas são diferentes.

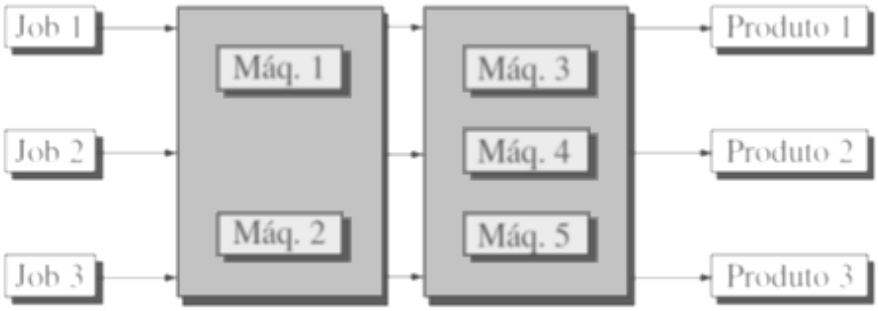

Figura 2.3 - Esquema do processo de execução de tarefas em máquinas paralelas

Os problemas de máquinas paralelas podem ser divididos em três grandes classes:

- Máquinas idênticas - em que o tempo de processamento de uma tarefa é igual em todas as máquinas;
- Máquinas uniformes - em que a cada máquina i está associada uma "velocidade" (por exemplo, se uma máquina é duas vezes mais rápida que outra, então todas as tarefas são, nessa máquina, processadas em metade do tempo);
- Máquinas não relacionadas - em que não há nenhuma relação entre as velocidades de processamento das diferentes máquinas. O problema de afetação de tarefas em estudo nesta tese pertence a esta classe.

2.2. Investigação operacional

A análise científica da utilização de recursos operacionais começou a ser estudada durante a segunda guerra mundial, para que os escassos recursos pudessem ser utilizados da maneira mais eficiente nas várias operações militares. Após o final da guerra, visto o sucesso obtido com a utilização da ferramenta, esta teve um grande e rápido avanço tanto nas empresas públicas como nas privadas, especialmente em Inglaterra e nos Estados Unidos, (Batalha, 2008).

Pidd (1998) e Cassel e Vaccaro (2007) definem a otimização como sendo um dos pilares da investigação operacional que dá suporte à tomada de decisão em ambientes produtivos. Portanto, esta ferramenta tem vindo a ser aplicada principalmente nas atividades de programação da produção e na gestão de cadeias de abastecimento. Por envolverem

geralmente ambientes complexos, com inúmeras variáveis e restrições, revelam-se o cenário ideal para a utilização das várias ferramentas disponíveis dentro desta área.

Silva *et al* (1998) define investigação operacional como um método científico de tomada de decisões. Em linhas gerais, consiste na descrição de um sistema organizado com o auxílio de um modelo, e através da experimentação com o modelo, na descoberta da melhor maneira de operar o sistema.

Para atingir os seus objetivos, o processo de decisão fundamentado na formulação e análise dos modelos matemáticos segue uma sequência lógica para que se atinjam os resultados exigidos. Silva *et al* (1998) define uma sequência a ser seguida na aplicação da investigação operacional à resolução de problemas:

1. Formulação do problema
2. Construção do modelo do sistema
3. Cálculo da solução através do modelo
4. Teste do modelo e da solução
5. Estabelecimento dos controlos da solução
6. Implementação e acompanhamento

Em seguida na Figura 2.4, e em forma de resumo do que é o processo de investigação operacional, apresenta-se um fluxograma genérico de uma pesquisa deste género

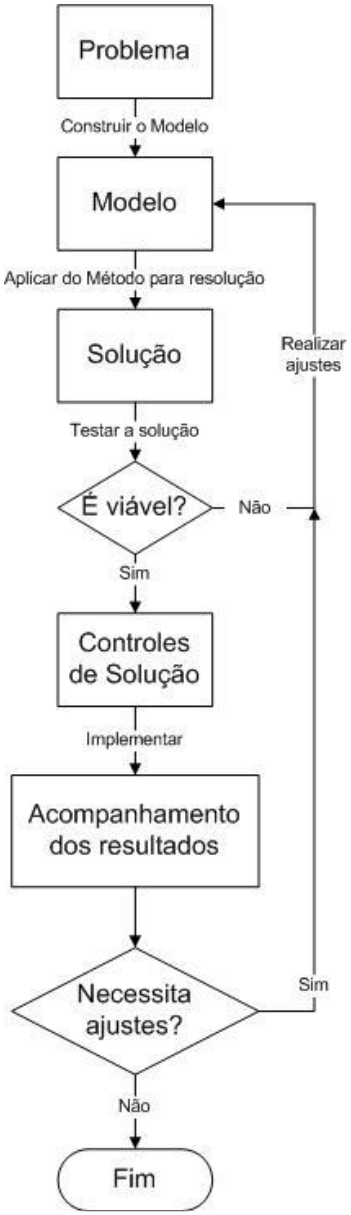

Figura 2.4 - Fluxograma de um processo de investigação operacional

2.2.1. **Modelação Matemática**

Segundo Goldbarg e Luna (2005), os modelos são representações simplificadas da realidade que preservam, para determinadas situações, uma equivalência adequada. O modelo matemático é uma representação simplificada (abstração) do problema real (Arenales *et al*, 2007).

A programação matemática trata de problemas de decisão em espaços de dimensões finitas e faz uso dos modelos matemáticos. As variáveis são definidas e as relações

matemáticas entre essas variáveis são estabelecidas de forma a descrever o comportamento do sistema (função objetivo e restrições), (Arenales *et al.*,2007).

2.2.1.1. Complexidade computacional

Em termos de dificuldade da resolução, os problemas de otimização combinatória são em geral classificados de acordo com a teoria da complexidade computacional. A distinção entre problemas NP-difíceis e problemas resolúveis em tempo polinomial implica a existência de um campo vasto de aplicação dos métodos enumerativos de soluções, Lawler (1983).

Como os problemas em estudo nesta tese são NP-Difíceis a abordagem mais comum para resolvê-los é através de métodos heurísticos, que serão apresentados em seguida.

2.2.1.2. Métodos aproximação

Segundo French (1982) os métodos de aproximação incluem aqueles em que se conhece quão próximas as suas soluções estão das ótimas e ainda uma variedade de outros métodos heurísticos que permitem obter boas soluções. Um método de aproximação "sub-óptimo" tenta apenas resolver, de uma forma aproximada, não tendo em vista a otimização de cada instância de um problema, não garantindo, portanto, que se encontre uma solução ótima (Morton, 1993). Embora as expressões "métodos heurísticos" e "métodos de aproximação" sejam muitas vezes utilizadas indistintamente na prática, eles refletem realidades diferentes, na medida em que no primeiro caso não se tem, normalmente, uma ideia clara de quão afastadas as soluções encontradas se encontram da solução ótima, no segundo caso é possível avaliar o grau de aproximação das soluções obtidas em relação à melhor das soluções possíveis (solução ótima).

2.3. Meta-Heurísticas

As técnicas heurísticas têm vindo a ser bastante utilizadas nos últimos anos, pois na maioria das situações reais encontrar uma boa solução em vez da melhor solução é o suficiente, e envolve muito menos esforço computacional e menos tempo despendido, são no fundo procedimentos que procuram por soluções ótimas, sem no entanto oferecerem a garantia de que encontrão uma.

Segundo Goldbarg e Luna (2005), uma heurística é uma técnica que procura alcançar uma boa solução utilizando um esforço computacional considerado razoável, sendo capaz de garantir a viabilidade ou a otimização da solução encontrada ou, ainda, em muitos casos, ambas, especialmente nas ocasiões em que essa procura parte de uma solução viável próxima do valor considerado ótimo.

As meta-heurísticas caracterizam-se basicamente por um processo de pesquisa de melhores soluções na vizinhança de uma solução, de acordo com regras e parâmetros previamente definidos, com o objetivo de encontrar um "ótimo" para o problema próximo do ótimo absoluto. A maior ou menor eficiência destes algoritmos depende do problema a otimizar, da estrutura da vizinhança definida para o problema, e dos procedimentos heurísticos utilizados na geração das soluções. De uma solução *v* do espaço de soluções deve ser sempre possível atingir qualquer outra solução em um número finito de passos, utilizando um determinado tipo ou tipos de movimentos.

As meta-heurísticas diferenciam-se entre si pelo mecanismo utilizado para escapar aos ótimos locais. Podem ser divididas em duas categorias, de acordo com o princípio usado para explorar o espaço de soluções: busca populacional e busca local.

Os métodos baseados em busca populacional consistem em manter um conjunto de boas soluções e combiná-las de forma a tentar produzir soluções ainda melhores. O principal exemplo deste tipo de meta-heurística são os Algoritmos Genéticos.

Nas meta-heurísticas baseadas em busca local, por sua vez, a exploração do espaço de soluções é feita por meio de movimentos, que são aplicados a cada passo sobre a solução corrente, gerando outra solução promissora na vizinhança desta. Alguns exemplos de meta-heurísticas baseadas neste princípio são a Busca Tabu (BT) e o *Simulated Annealing* (SA).

2.3.1. **Simulated annealing**

Esta abordagem foi originalmente proposta, por por Kirkpatrick, Gelatt e Vechi, em 1983 por analogia com o processo de recozimento (*annealing*) de um sólido.

De acordo com Izquierdo (2000), o S*imulated Annealing* é um método iterativo, que combina técnicas de busca local e aleatoriedade, procurando assim, quando aplicado a um problema de otimização, evitar a estagnação num ótimo local.

A meta-heurística *Simulated Annealing* baseia-se num modelo original da física, o do recozimento de metais. Neste processo, o arrefecimento gradual a partir de uma alta

temperatura de um material leva-o a estados mínimos de energia. Comparando-se o modelo físico com o matemático, vê-se que sob altas temperaturas as partículas estão completamente desorganizadas, correspondendo a uma configuração aleatória de um problema de otimização. Pequenas e graduais alterações nas posições de algumas destas partículas resultam numa variação de energia, ou seja, uma alteração no valor da função objetivo (Pozzo, 2010). A Tabela 2.1 demonstra as principais similaridades entre o modelo físico e um problema de otimização.

***Annealing* de um Sólido (Recozimento)**	**Problema de Otimização Combinatória**
Estado físico	Solução
Energia do estado	Valor da função objetivo
Temperatura	Parâmetro de controlo
Prescrição de arrefecimento	Regras que regem a inicialização e o decréscimo do parâmetro de controlo
Estado fundamental do sólido (congelamento)	Ótimo global
Arrefecimento rápido	Algoritmo de busca local
Arrefecimento lento e controlado	Algoritmo *Simulated annealing*

Tabela 2.1 - – Comparação entre o modelo físico (*Annealing*) e um modelo de otimização

O objetivo de um método desta natureza é encontrar a melhor solução entre um número finito de soluções possíveis. A técnica de *Simulated Annealing* é uma técnica particularmente atrativa pois permite encontrar soluções próximas da ótima à custa de um esforço computacional pouco exagerado. É de notar que neste tipo de método, não é, geralmente, possível saber se a melhor solução encontrada é o ótimo global. Esta característica restringe a utilização deste tipo método a casos em que um bom ótimo local é aceitável (Nurmela, 1993).

Os métodos baseados em *Simulated Annealing* são métodos que têm apresentado excelentes resultados na resolução de muitos problemas de otimização combinatória, do tipo NP-difíceis. Estes métodos podem ser entendidos como uma extensão dos métodos de pesquisa local probabilística simples. Se a nova solução proposta é igual ou melhor do que

a solução atual então esta será aceite. Se a nova solução proposta é pior do que a solução atual poderá, mesmo assim, ser aceite mediante uma determinada probabilidade de aceitação. Idealmente, quando a PL atinge um ótimo local "fraco", esta técnica permite abandonar esse ótimo local.

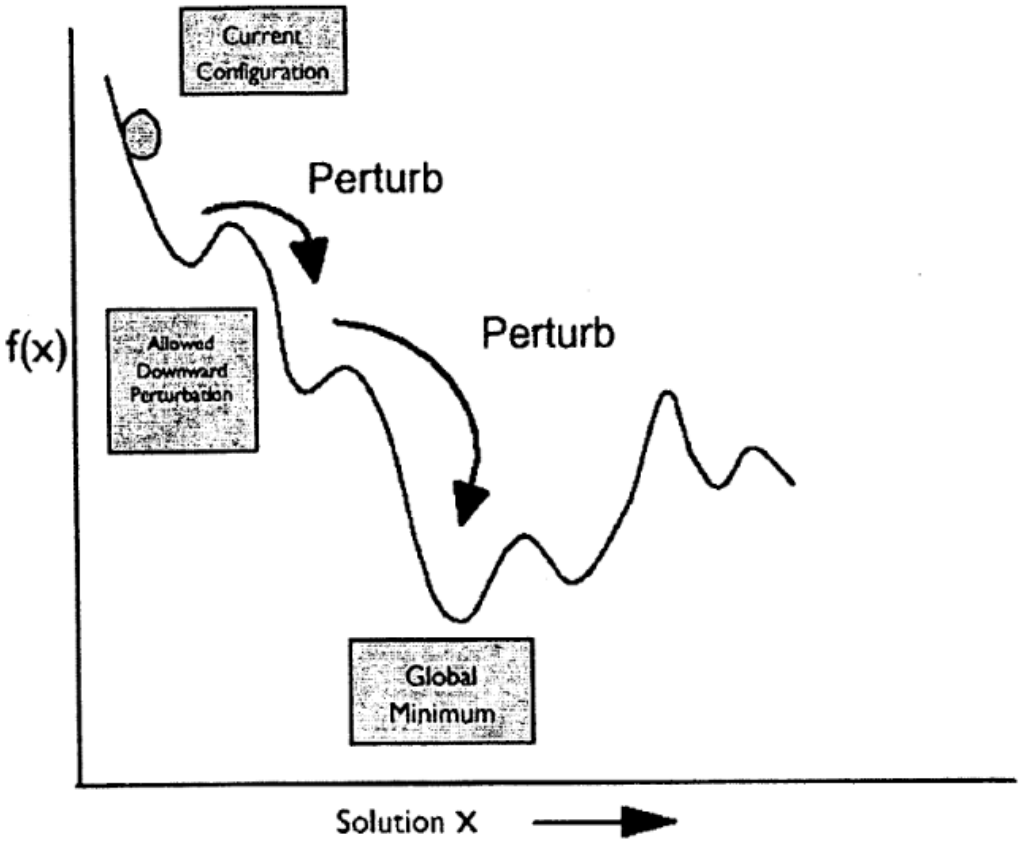

Figura 2.5 - Processo de busca da melhor solução utilizando o *Simulated Annealing* (Aschakov, 2011)

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

Esta técnica gera um caminho Markoviano, em que o sucessor de um ponto atual é escolhido estocasticamente, com uma probabilidade que depende apenas do ponto atual e não da história prévia da pesquisa. A propriedade não Markoviana é uma bênção misturada, ou seja, permite resultados heurísticos, que são bastante bons em muitos casos, mas torna a análise teórica do método difícil. (Varela,M., Revista de Investigação Operacional, 2001).

Na Tabela 2.2 apresentam-se as vantagens e desvantagens do *Simulated Annealing* em comparação como os outros processos de otimização.

Vantagens	Desvantagens
• Implementação simples, já que visita somente uma solução a cada iteração, e basta calcular o valor da função objetivo neste ponto; • Consegue lidar com modelos altamente não lineares, dados caóticos e muitas restrições; • Bastante robusto.	• Apesar de convergir para a solução ótima, a velocidade de redução de temperatura exigida implica visitar um número exponencial de soluções; • A princípio é necessário um processo lento de arrefecimento e isso resulta em tempos elevados de processamento.

Tabela 2.2 – Comparação entre o *Simulated Annealing* e os outros processos otimização

2.3.1.1. Parâmetros do *Simulated Annealing*

Em seguida, Figura 2.6, apresentar-se-á um algoritmo base de uma pesquisa Simulated Annealing e em seguida serão introduzidos os parâmetros que devem ser definidos antes do início da pesquisa pois são necessários ao bom desempenho do algoritmo.

```
Algoritmo SimulatedAnnealing();
Início
    LerParamet();
    LerDados();
    IniciarT0();
    n = 1
    x1 = SeqInicial();
    N(xn) = GerVizinhança(xn);
    x* = x1
    F* = F(x1);
    Enquanto Criterioparagem < NumeroIterações
        Para i=1 até L
            x = SoluçãoAleatória(N(xn));
            Se F(x) < F(xn) então
                xn = x
                Se F(x) < F* então
                    x* = x
                    F* = F(x)
                    Criterioparagem=0
                Senão
                    Criterioparagem= Criterioparagem+1
                FimSe
                xk* = xn+1
                Fk* = F(xk*)
            Senão
                p = aleatorio();
                pn = calc_pn();
                Se p < pn então
                    xn+1 = x
                FimSe
            FimSe
            n = n+1
            N(xn) = GerVizinhança(xn);
            Tn+1 = α Tn
        FimPara
    FimEnquanto
Fim
```

Figura 2.6 - Algoritmo *Simulated Annealing* (Madureira,1995)

Com o objetivo de tornar o algoritmo operacional devem ser tomadas algumas decisões genéricas, tais como: a forma de definir a temperatura inicial, a escolha da probabilidade de aceitação de soluções piores que a melhor encontrada, o esquema de

arrefecimento, o número de iterações que são realizadas à mesma temperatura e o critério de paragem.

a) Definição da temperatura inicial

A temperatura inicial T_0 de uma sequência de arrefecimento geométrico é geralmente determinada de tal forma que a probabilidade inicial de aceitação de "maus movimentos", ou soluções piores, seja aproximadamente igual a um valor $Prob_0$. É usual calcular a temperatura a que corresponde uma probabilidade de aceitação que varia entre 0.4 e 0.5.

Deve-se propor um valor alto para o parâmetro T_0, e em seguida deve-se fazê-lo decrescer em cada iteração, ou a cada tempo *k* com um determinado padrão de decréscimo. Desta forma, o critério de aceitação da nova solução é função da temperatura *T* e da quantidade de "energia" despendida (diferença no valor da função objetivo) a atingir.

b) Probabilidade de aceitação

Por analogia com a termodinâmica, é usual escolher-se para esta meta-heurística a distribuição de Boltzman:

$$\text{prob(n)} = exp\left(-\frac{\Delta \text{Fn}}{T_n}\right); \tag{2.1}$$

Em que a temperatura na iteração n é dada por:

$$\Delta F_n = F(X) - F(x_n) \text{ e } T_n . \tag{2.2}$$

c) A escolha do esquema de arrefecimento

A temperatura T_n diminui com o tempo de forma a excluir, no fim, os movimentos "maus"(a probabilidade de aceitação tende para 0). Começando em T_0 a temperatura é mantida constante em L iterações consecutivas.

A temperatura evolui segundo uma progressão geométrica, diminuindo pela multiplicação por um fator constante α ($0<\alpha<1$) após cada série de L iterações.

Após kL passos, a temperatura é :

$$T_{kL}=T_k=\alpha^{k}T_{0,} \tag{2.3}$$

Ou, de outra forma;

$$T_{k+1}=\alpha\, T_{k,} \tag{2.4}$$

Normalmente, tomam-se valores para α entre 0.9 e 0.99.

T_0 = Temperatura inicial

α = taxa de arrefecimento

L = comprimento do "degrau"

d) Número de iterações à mesma temperatura, L_n

A capacidade do algoritmo explorar convenientemente a vizinhança depende fundamentalmente do número de iterações que se realizam à mesma temperatura T_n. É habitual definir-se L_n=L, isto é, constante e independente da temperatura, e diretamente relacionado com o tamanho da vizinhança e as características particulares do problema em questão.

e) Escolha de um critério de paragem

Existem diversos critérios possíveis de serem utilizados como critério de paragem mas o mais comum e o que será utilizado neste estudo é aquele em que o procedimento é terminado quando a função objetivo não for melhorada num número específico (a definir consoante as variáveis do problema em estudo) seguido de iterações.

3. DESCRIÇÃO DOS PROBLEMAS

Nesta tese, abordam-se dois problemas diferentes relativos ao planeamento da produção. O primeiro é um problema de sequenciamento de tarefas em máquina única ao passo que o segundo caso consiste num problema de afetação de tarefas em 2 máquinas paralelas.

3.1. Problema 1 – Sequenciamento em máquina única

No primeiro problema, existem n tarefas para serem executadas, numa única máquina, com tempos de processamento, datas de entrega diferentes e por vezes pesos, diferentes. O objetivo passa, nos problemas com 5 (T5) e com 19 (T19) tarefas, por minimizar o atraso total, que se caracteriza pela soma dos atrasos na execução de todas as tarefas, ao passo que para os problemas WT40, WT50 e WT100 ("*weighted tardiness*"), passa por minimizar o atraso pesado que representa o somatório do produto dos pesos de cada tarefa pelos seus atrasos respetivos.

Há uma correspondência entre a sequência das n tarefas uma-a-uma e uma permutação dos índices dos trabalho 1, 2, ..., n. O número total de soluções distintas para este problema básico de sequenciamento é dado por n!, que é o número de permutações possíveis dos n elementos. Não se irão testar todas as soluções como foi referido anteriormente e por isso é necessário definir o problema convenientemente para que das soluções (sequencias) testadas seja encontrada uma que possa ser considerada uma boa solução. Definiram-se as seguintes características para o conjunto destes problemas:

3.1.1. Caraterísticas das Tarefas

- As tarefas são independentes e caracterizadas por tempos de processamento;
- Todas as tarefas a executar estão disponíveis para serem executadas no instante 0;
- São conhecidas previamente as características das tarefas, nomeadamente tempos de processamento e datas de entrega;
- Para os problemas T5 e T19 não há ordem de prioridade para as tarefas e todas têm o mesmo grau de importância;
- Nos problemas WT40, WT50 e WT100 todas as tarefas têm associado um peso (w), que representa o grau de prioridade de execução de cada tarefa;

- Não existem tempos de *setup* entre a realização das tarefas;

3.1.2. Características da Máquina

- No instante 0 a máquina está disponível para o processamento de um conjunto de n tarefas independentes;
- A máquina está continuamente disponível e nunca existem tempos de espera entre o fim da execução de uma tarefa e o início do processamento da tarefa seguinte;
- Uma vez que a máquina inicia o processamento da tarefa, esta é processada até ao fim, sem interrupção;

3.1.3. Características do Processamento das tarefas

- O tempo de processamento de cada operação é dado no início e permanece constante, independentemente da ordem de execução;
- O tempo de preparação e de transporte é incluído no tempo de processamento de uma operação;
- O Peso de cada tarefa é definido inicialmente e permanece estável durante todo o processo;

3.1.4. Funções Objetivo

3.1.4.1. Atraso total

Cada tarefa j requer um tempo de processamento e tem uma data de entrega especificada. O atraso total representa a diferença entre o instante em que a tarefa termina de ser executada e a data de entrega da tarefa, especificada inicialmente.

O objetivo passa por encontrar uma sequência de processamento para as n tarefas, de tal modo que seja minimizado o atraso total

3.1.4.2. Atraso Pesado ("*Weighted Tardiness*")

O atraso pesado representa o somatório do produto dos pesos associados a cada tarefa e do atraso associado a cada uma delas. O atraso representa a diferença entre o instante em que a tarefa termina de ser executada e a data de entrega da tarefa, especificada inicialmente.

3.2. Problema 2 – Afetação de tarefas em Máquinas paralelas

No segundo problema existem n tarefas para serem executadas, em duas máquinas que trabalham em paralelo, o tempo de processamento varia consoante a máquina em que a tarefa é executada. O objetivo é minimizar o *Makespan*, ou seja o tempo total necessário para executar as n tarefas. Definiram-se as seguintes características para este problema:

3.2.1. Características das tarefas

- Cada máquina só processa uma tarefa de cada vez;
- A execução de uma tarefa numa máquina não pode ser interrompida;
- O tempo de execução para cada tarefa depende da máquina que a executa.
- O processamento de qualquer tarefa não pode ser feito por mais do que uma máquina
- Não serão considerados tempos de *setup*;

3.2.2. Características das Máquinas

- As máquinas podem executar o mesmo tipo de funções (máquina s paralelas), ou seja, todas as máquinas podem executar todas as tarefas;
- O tempo de execução para cada tarefa depende da máquina que a executa;
- Não há definição de prioridades entre as tarefas e todas têm o mesmo grau de importância;
- As duas máquinas estão disponíveis para iniciar o processamento das tarefas no instante 0;
- Qualquer máquina pode processar qualquer tarefa;
- Uma máquina não pode executar mais do que uma tarefa de cada vez;

3.2.3. Características do processamento das tarefas

- O tempo de processamento de cada tarefa é dado no início e permanece constante, independentemente da ordem de execução.
- O tempo de preparação e de transporte é incluído no tempo de processamento da tarefa.

3.2.4. **Função Objetivo**

3.2.4.1. Makespan

É o critério de otimização mais considerado em problemas de sequenciamento de tarefas e visa uma melhor utilização das máquinas o que equivale a menores perdas (Pinedo, 2008).

A abordagem a este problema tem como objetivo minimizar o *makespan* ou seja, o instante de conclusão da última tarefa a ser executada pelo conjunto das máquinas.

4. MEDIDAS DE DESEMPENHO

4.1. Problema 1 - Sequenciamento em máquina única

Ao lidar com atributos de tarefas para o modelo de uma única máquina, é útil distinguir entre informações conhecidas previamente e informações que são geradas como resultados de decisões de sequenciamento. A informação que é conhecida previamente serve como parâmetro de entrada para função de sequenciamento. As duas informações básicas que ajudam a descrever trabalhos no problema básico determinístico de uma única máquina, e que já foram minuciosamente discutidas no capítulo anterior deste trabalho, são:

- Tempo de processamento (p_j): tempo de processamento requerido pelo trabalho j;
- Data de entrega (d_j): é o instante em que o processamento do trabalho j está prometido para ser concluído.

Das informações que são calculadas como resultado das decisões das diferentes sequências geradas as que mais interessam neste estudo são:

- Tempo de conclusão (C_j) – instante de tempo em que o processamento da tarefa j é concluída;

$$C_j = t_j + p_j \tag{4.1}$$

- Atraso (lateness, L_j) – Diferença entre o instante de conclusão e a data prometida da tarefa j, podendo assumir valores positivos ou negativos. É importante notar que a quantidade em atraso toma valores negativos sempre que uma tarefa é completada mais cedo do que a respetiva data de entrega. Um atraso negativo representa melhor serviço que o requerido.

$$L_j = C_j - d_j \tag{4.2}$$

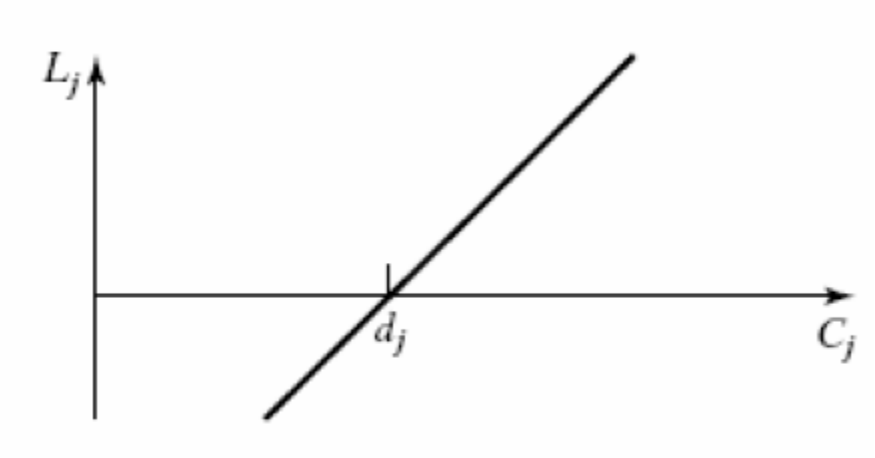

Figura 1.1 - Gráfico do Atraso (L_j) x Tempo de Conclusão (C_j) (Pinedo, 2008)

- Atraso positivo (tardiness, T_j) - representa um atraso na conclusão de processamento de uma tarefa; sendo por isso referido como atraso. É quanto a tarefa j é concluída com atraso em relação à sua data prometida, caso contrário será considerado zero

$$T_j = Max\{0; L_j\} \tag{4.3}$$

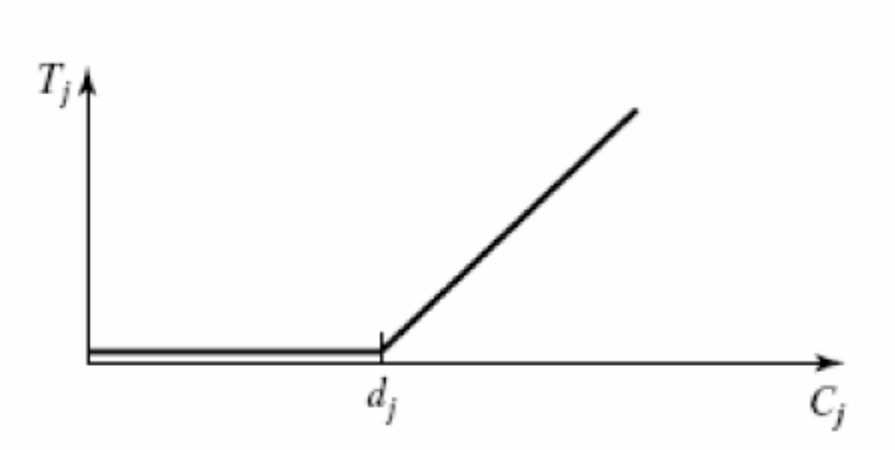

Figura 4.2 - Gráfico (T_j) x (C_j) (Pinedo,2008)

- Atraso total, T_t - Esta é uma das medidas de desempenho que serão avaliadas neste trabalho sob a forma de função objetivo de algumas variantes do problema 1

$$T_t = \sum_{j=1}^{n} T_j \tag{4.4}$$

- Atraso pesado (Weighted Tardiness), W_t - Esta é uma das medidas de desempenho que serão avaliadas neste trabalho sob a forma de função objetivo de algumas variantes do problema 1

$$T_t = \sum_{j=1}^{n} T_j.W_j \tag{4.5}$$

4.2. Problema 2 - Afetação de tarefas em Máquinas paralelas

Como se verifica na abordagem do problema 1 também no problema 2, que relaciona duas máquinas paralelas não idênticas, é útil distinguir entre informações conhecidas previamente e informações que são geradas como resultados de decisões do algoritmo para melhoria da afetação das tarefas. A informação que é conhecida previamente serve como parâmetro de entrada para função de afetação de tarefas. Ao contrário do primeiro problema estudado, neste apenas temos um tipo de informação inicial denominada de tempo de processamento:

Tempo de processamento, p_{ij} : tempo para processamento da tarefa *j* na máquina *i*.

Das informações que são calculadas como resultado das decisões das diferentes soluções geradas a que se revela determinante para estudarmos a função objetivo, *makespan*, é o tempo de conclusão:

- Tempo de conclusão, c_{ij}: instante de conclusão da tarefa *j* na máquina *i*.

$$C_{ij}=t_{ij}+p_{ij} \tag{4.6}$$

A medida de desempenho que será avaliada para o problema 2 é o *makespan*.

- *Makespan*, C_{max} – O *makespan* é definido como o maior dentre as datas de conclusão (C_1,.., C_n), ou seja, o tempo de conclusão do último trabalho a sair do sistema ($C[_n]$). Minimizar o *makespan* normalmente implica uma boa utilização dos recursos (máquinas).

$$C_{max} = Max(C1, C2) \tag{4.7}$$

5. IMPLEMENTAÇÃO COMPUTACIONAL

Partindo do algoritmo básico do *Simulated Annealing*, descrito anteriormente estruturaram-se 2 algoritmos baseados nesta meta-heurística mas adaptados aos dois problemas em estudo neste trabalho, a saber:

- Sequenciamento em máquina única
- Afetação de tarefas em máquinas paralelas

5.1. Estruturas de dados

No programa para o problema de sequenciamento em máquina única (problema 1), criou-se apenas um *array* (n x 4) onde cada posição representa uma estrutura constituída pelo número da tarefa, o tempo de processamento respetivo, a data prevista para entrega e, no caso dos problemas WT, pelo peso de cada tarefa.

No programa para o problema de afetação de tarefas em máquinas paralelas foram criados 3 *array*, um (n x 2) em que no primeiro eixo se situam todas as tarefas (n) e no segundo se guardam as velocidades do processamento das n tarefas em cada uma das máquinas. Foram criados ainda outras duas estruturas deste género, uma para cada máquina, em que cada *array*, com tamanho (n x 1), possui o número das tarefas afetas a cada máquina.

5.2. Problema 1 – Sequenciamento em máquina única

1. Ler os dados do problema
    a. Número de tarefas (n)
    b. Tempos de processamento, datas de entregas das n tarefas e peso caso se trate de um problema que considere esta variável
2. Sortear duas tarefas
3. Trocar posição das tarefas
4. Determinar o valor da função objetivo (atraso total) para a nova solução
5. Verificar se o valor da Função objetivo é melhor que o até agora encontrado
    a. Se sim, aceitar

   b. Se não, calcular a probabilidade de aceitação da nova solução
6. Aceitar ou não a nova solução.
7. Verificar se o critério de paragem é satisfeito
   a. Se sim, terminar a procura
   b. Se não, T= r x T
8. Se $T > \delta$ voltar ao passo 1
9. Fim

Resumidamente, o algoritmo do *Simulated Annealing* aplicado ao Problema 1 pode ser dividido essencialmente em 6 fases principais:

- Definição dos dados do *Simulated Annealing*
- Leitura dos dados do problema
- Seleção aleatória das tarefas a trocar
- Troca das tarefas selecionadas
- Cálculo do valor da função objetivo para a nova sequência
- Aceitação ou não da sequência em estudo

Nesta fase do trabalho descrever-se-á o código relacionado com o algoritmo desenvolvido para resolver os problemas que consideram o peso como parte da função objetivo (*Weighted Tardiness*) porque todos os programas são parecidos e este acaba por ser o mais complexo, pelo que percebendo este torna-se fácil perceber o funcionamento dos outros.

Definição dos parâmetros do *Simulated Annealing*

Esta fase baseia-se na definição dos dados relativos ao algoritmo de *Simulated Annealing* propriamente dito, ou seja é a definição dos parâmetros necessários a aplicação da meta-heurística, mais propriamente, a temperatura inicial (T_0), a temperatura final (T), o rácio de arrefecimento, r, e o tamanho do intervalo que define o número de iterações que se efetuam à mesma temperatura, L. A temperatura inicial e a temperatura final, são calculadas isoladamente e com recurso a muita experimentação, para cada problema, pois são parâmetros reguladores da pesquisa e são influenciados pelo número de tarefas e tipo de problema.

```
Dim Temp As Double
    Temp = 250

Dim r As Double
    r = 0.9

Dim Tmin As Double
    Tmin = 0.01

L = 0

    Do Until L = 5
```

Leitura dos dados do problema

Nesta fase, ocorre a criação de um *array* (MyArray) com a informação relativa à sequência de tarefas inicial e aos tempos de processamento, datas de entrega e peso de cada tarefa. Este *array* recolhe a informação dos dados presentes na folha de Excel. É portanto criado um *array* bidimensional (19 x 4) em que cada elemento (tarefa) tem as dimensões 1x4 (número da tarefa, tempo de processamento (P), data de entrega (d) e peso (w)).

```
MyArray = Sheet1.Range("B3:E42").Value
```

Geração das soluções vizinhas

- Seleção aleatória das tarefas a trocar,

Nesta fase do algoritmo são escolhidas de forma aleatória duas tarefas (trf1 e trf2) das n tarefas, caso a escolha recaia sobre a mesma tarefa, o sorteio da segunda tarefa volta a ser feito até que a tarefa escolhida, trf2, seja diferente da trf1.

```
Trf1 = Int((l1 - 0 + 1) * Rnd + 1)
Trf2 = Int((l1 - 0 + 1) * Rnd + 1)

If Trf2 = Trf1 Then

    While Trf2 = Trf1

    Trf2 = Int((l1 - 1 + 1) * Rnd + 1)
    Trf2 = Trf2 + 1
    Beep
    Wend

End If
```

- Troca das tarefas selecionadas,

Em seguida procede-se à troca de posição entre as duas tarefas no *array* que representa a sequência de processamento.

```
Dim tmp
For i = 1 To 4
    tmp = MyArray(Trf1, i)
    MyArray(Trf1, i) = MyArray(Trf2, i)
    MyArray(Trf2, i) = tmp
Next
```

Avaliação das soluções

- Cálculo do valor da função objetivo

Nesta fase calcula-se o valor da função objetivo. Começa por calcular o instante, i, em que cada tarefa é dada como terminada, e em seguida é calculado o atraso, através da diferença entre a data prevista para entrega, d, e o instante, i. Por fim o valor da função objetivo é calculado através do somatório das multiplicações dos valores dos atrasos pelo valor do peso, w, correspondente a cada tarefa.

```
Instante(1) = MyArray(1, 2)

For i = 2 To n

    Instante(i) = Instante(i - 1) + MyArray(i, 2)

Next

For i = 1 To n

Atraso(i) = Instante(i) - MyArray(i, 3)
    If Atraso(i) < 0 Then
    Atraso(i) = 0
    End If
 Next


 For i = 1 To n
 Atrasow(i) = 0
 Next

 For i = 1 To n
 Atrasow(i) = Atrasow(i) + Atraso(i) * MyArray(i, 4)
 Next
```

- Aceitação ou não da solução (sequência) em estudo.

Neste momento da execução do programa a função objetivo da solução em estudo é comparada com o melhor valor da função objetivo (atrasos pesados) encontrado até este momento, caso a solução seja melhor (menor) é aceite, guardada e passa a ser a solução atual do problema caso não seja melhor é calculada uma probabilidade de aceitação e sorteado um numero entre 0 e 100. Caso o número sorteado seja inferior ao valor da probabilidade de aceitação "atual" a solução é aceite e passa a ser a solução em estudo, caso contrário a solução é desprezada.

O algoritmo termina quando o critério de paragem for atingido, ou seja, quando o valor da função objetivo não for melhorado após um número de iterações consecutivas.

```
prob = Exp(((MelhorAtraso - atrasoaux)) / Temp)
        calha = Int((100 - 0 + 1) * Rnd + 0)
        calha2 = calha / 100

If prob > 1 Then
prob = 1
End If

    If atrasoaux < MelhorAtraso Then

        MelhorAtraso = atrasoaux
        AtrasoSA = atrasoaux
        Sheet3.Range("B3:E42").Value = MyArray
        Sheet1.Range("B3:E42").Value = MyArray

        Sheet3.Range("H10") = MelhorAtraso


        Application.ScreenUpdating = True

        fim = 0

    ElseIf calha2 < prob Then

            Sheet1.Range("B3:E42").Value = MyArray
            AtrasoSA = atrasoaux

            Application.ScreenUpdating = True

        fim = fim + 1

    End If
```

5.3. Problema 2 - Afetação de tarefas em máquina paralelas

A diferença fundamental entre os dois algoritmos prende-se com o facto de que enquanto no algoritmo referente ao Problema 1, que visa solucionar um problema de sequenciamento em máquina única, a procura de diferentes soluções se baseia na troca da posição na sequência entre duas tarefas, no algoritmo afeto ao Problema 2 a busca por novas soluções é feita com recurso a dois tipos possíveis de alteração nas soluções, a saber: troca (bidirecional) de tarefas entre as duas máquinas ou passagem (unidirecional) de tarefas afetas a uma máquina para a outra.

1. Ler os dados do problema
    a. Número de tarefas (n)
    b. Número de máquinas (m) e diferentes tempos de processamento
2. Calcular o valor da função objetivo para a distribuição de tarefas inicial
3. Gerar um número aleatório (rnd) entre 0 e 99
    a. Se rnd <49, ir para passo 3
    b. Se rnd >=49, ir para passo 4
4. Aleatoriamente selecionar duas tarefas (T1 e T2), uma de cada máquina
    a. Trocar posição das duas tarefas selecionadas
5. Atribuir uma tarefa de uma máquina à outra
    b. Selecionar uma máquina, 1 ou 2 (MS)
    c. Selecionar uma tarefa da máquina selecionada (TS)
    d. Transferir a tarefa TS, da máquina MS para a outra máquina
6. Determinar o valor da função objetivo para a nova solução
7. Verificar se o valor da Função objetivo (*Makespan*) é melhor que o até agora encontrado
    e. Se sim, aceitar
    f. Se não, calcular a probabilidade de aceitação da nova solução
8. Aceitar ou não a nova solução.
9. Verificar se o critério de paragem é satisfeito
    g. Se sim, terminar a procura
    h. Se não, T= r x T

10. Se $T > T_0$ voltar ao passo 3
11. Fim

Resumidamente, o algoritmo do *Simulated Annealing* aplicado ao Problema 2 pode ser dividido essencialmente em 7 fases principais,

- Definição dos dados do *Simulated Annealing*
- Leitura dos dados do problema
- Seleção do tipo de alteração a efetuar na solução atual (troca ou passagem de uma tarefa de uma máquina para a outra)
- Seleção aleatória das tarefas a trocar ou seleção da tarefa a ser transferida de uma máquina para a outra e o sentido da transferência
- Troca ou transferência da(s) tarefa(s) selecionada(s)
- Cálculo do valor da função objetivo para a nova sequência
- Aceitação ou não da sequência em estudo

Dadas as similaridades existentes entre o código para o problema 1 e o código para o problema 2 nesta fase do trabalho apenas se vai olhar com maior detalhe para uma das 7 fases deste algoritmo/código, pois todas as outras fases, no essencial, são idênticas às fases do algoritmo acima descrito.

- Seleção do tipo de alteração a efetuar na solução atual e da(s) tarefa(s) a transferir ou trocar

O código para esta fase do processo pode ser consultado no anexo 1, pois esta é algo complexa e o código é bastante extenso.

Esta fase do programa começa por sortear um valor (acção) entre 0 e 99.

Caso o valor sorteado seja inferior a 49 procede-se a uma troca entre tarefas da máquina 1 e da máquina 2. Em seguida são sorteadas duas tarefas, uma de cada máquina, aleatoriamente trf1 e trf2, e é efetuada a troca entre as duas máquinas.

Caso o valor sorteado (acção) seja igual ou superior a 49, o programa averigua se alguma das máquinas possui, naquele momento, 2 tarefas afetas, caso ambas as máquina tenham mais que 2 tarefas afetas o programa sorteia uma máquina (1 ou 2) e em seguida

seleciona, aleatoriamente, uma tarefa das que estão afetas a essa máquina (trf1 ou trf2) e procede à transferência dessa tarefa que passa a estar afeta à máquina não selecionada.

Caso o valor sorteado seja igual ou superior a 50 mas exista uma máquina que naquele momento possua apenas 2 tarefas a si afetas, não ocorre sorteio da máquina e a transferência da tarefa (escolhida aleatoriamente) é sempre efetuada da máquina que possui mais tarefas para a que possui menos, que terá sempre duas tarefas, pois este é o mínimo de tarefas afeto a cada máquina aceite por este programa.

6. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Para uma melhor análise dos resultados decidiu-se, para cada problema realizar 10 corridas do algoritmo *Simulated Annealing*, para que apesar do carácter aleatório inerente a todas as meta-heurísticas os resultados pudessem ser considerados credíveis. Tanto para os problemas de sequenciamento afetos ao primeiro algoritmo,T5, T19, WT40, WT50 e WT100, como para os problemas de afetação de tarefas a serem resolvidos pelo segundo, M6 não relacionados, M6 uniforme e M6 iguais, para que a análise fosse o mais detalhada possível determinaram-se os seguintes parâmetros para um universo de 10 corridas:

- Média dos valores da função objetivo
- Desvio Padrão
- Percentagem de desvio padrão
- Melhor (Menor) valor obtido
- Pior (Maior) valor obtido
- Desvio do valor médio

E em relação à melhor solução encontrada para cada problema:

- Nº Soluções avaliadas
- Tempo de execução
- Comparação com Valor ótimo em valor e em percentagem

"As meta-heurísticas caracterizam-se basicamente por um processo de pesquisa de melhores soluções na vizinhança de uma solução, de acordo com regras e parâmetros previamente definidos, com o objetivo de encontrar um "ótimo" para o problema próximo do ótimo absoluto. A maior ou menor eficiência destes algoritmos depende do problema a otimizar, da estrutura da vizinhança definida para o problema, e dos procedimentos heurísticos utilizados na geração das soluções. De uma solução *v* do espaço de soluções deve ser sempre possível atingir qualquer outra solução em um número finito de passos, utilizando um determinado tipo ou tipos de movimentos."

Ao analisar os resultados computacionais obtidos depois das várias soluções testadas, constata-se desde logo que as soluções encontradas, estão sujeitas e espelham o caráter

aleatório próprio das meta-heurísticas, pelo que, qualquer que seja o problema ou o método de pesquisa de soluções utilizado é sempre necessário realizar mais que uma corrida, para não correr o risco de assumir como soluções finais valores que decorrem de soluções extremas devolvidas pelo algoritmo. Posto isto, constata-se desde logo que, e devido ao já falado grau de aleatoriedade na geração das "sub-vizinhanças" deste algoritmo se torna difícil tirar conclusões absolutas e definitivas quanto aos valores obtidos, pois ao analisar de uma forma global os resultados obtidos constata-se que em sensivelmente 1 em cada 10 corridas ocorre uma convergência prematura que acaba por resultar em ótimos locais que não refletem o verdadeiro potencial destes algoritmos para obtenção de "boas soluções."

Os dados presentes nas tabelas 6.1 e 6.2 são dados puramente estatísticos que mais do que servirem para avaliar a qualidade das soluções servem sobretudo para perceber a variação e a coerência das soluções finais obtidas. Assim, e atentando agora nestas tabelas percebe-se, que, como seria de esperar há uma maior discrepância dos valores das melhores soluções obtidas no conjunto das dez corridas nos problemas com maior número de tarefas.

Nº de Tarefas (n)	*T5(n=5)*	*T19(n=19)*	*WT40 (n=40)*	*WT50 (n=50)*	*WT100 (n=100)*
Média de valor da função objetivo	370	22,5	89021,1	31804,4	16841,8
Desvio Padrão	0	5,4	9485,8	8244,5	8296,5
Desvio Padrão em Percentagem	0	24%	10,66 %	25,92%	49,26%
Maior valor obtido	370	57,1	100302	43132	37615
Menor valor obtido	370	10,4	74075	15493	10496

Tabela 6.1 – Resultados da aplicação do algoritmo aos problemas de sequenciamento em máquina única

Nºtarefas	*n=6, m=2 (não relacionadas)*	*n=6, m=2 (uniformes)*	*n=6, m =2 (iguais)*
Média de valor da função objetivo	22	24	36
Desvio Padrão	0	0	0
Maior valor obtido	22	24	36
Menor valor obtido	22	24	36

Tabela 6.2 – Resultados da aplicação do algoritmo aos problemas de afetação de tarefas em máquinas paralelas

No problema T5 para o sequenciamento e nos 3 problemas do tipo M6 para afetação de tarefas, os valores não sofrem qualquer desvio, tendo-se verificado a mesma solução final que corresponde à solução ótima, para todas as corridas realizadas. Esta coerência deve-se ao facto do número de soluções possíveis ser relativamente pequeno e por essa razão o programa conseguir em muito pouco tempo avaliar grande parte das soluções possíveis. Por exemplo, para o problema T5 em que existem 5 tarefas, o número de soluções possíveis é 120 (5x4x3x2x1x) e o programa testou 100 dessas soluções em 1,2 segundos. Nos problemas de afetação de tarefas em máquinas paralelas a tendência também foi esta, por exemplo na solução encontrada para o problema com 6 tarefas em 2 máquinas não relacionadas o programa avaliou 87 soluções possíveis em menos de 1 segundo, como se pode verificar nos resultados espelhados na Tabela 6.3.

Assim pode-se concluir que para problemas pequenos, quer sejam de sequenciamento em máquina única quer sejam de afetação de tarefas em máquinas paralelas o *Simulated Annealing* devolve a solução ótima, bastante rapidamente.

Nºtarefas	*n=6, m=2(não relacionadas)*	*n=6, m=2 (uniformes)*	*n=6, m=2 (iguais)*
Melhor/Menor valor obtido	22	24	36
Nº Soluções avaliadas	87	95	102
Tempo execução	0,99 s	1,05 s	1,07 s
Valor ótimo de referência	22	24	36
Diferença entre o ótimo e o melhor obtido neste estudo	0	0	0
Desvio do valor ótimo	0%	0%	0%

Tabela 6.3 – Resultados da aplicação do algoritmo aos problemas de afetação de tarefas em máquinas paralelas

Olhando agora para os outros problemas resolvidos, com maior número de tarefas, percebe-se facilmente que o valor do desvio padrão aumenta com o número de tarefas dos problemas, à exceção do problema T19 que em comparação com a tendência que seguem os outros apresenta uma percentagem de desvio padrão alta. Este facto pode-se dever a uma casualidade relacionada com a amostra recolhida, uma vez que seria de esperar que a percentagem de desvio padrão fosse menor que para os problemas com 40, 50 e 100 tarefas.

Esta tendência do crescimento do desvio padrão com o número de tarefas está relacionado com a percentagem de soluções avaliadas, que como é óbvio é menor quanto maior for o número de tarefas e consequentemente maior o número de soluções possíveis.

Nº de Tarefas (n)	*T5(n=5)*	*T19(n=19)*	*WT40 (n = 40)*	*WT50 (n = 50)*	*WT100 (n = 100)*
Melhor/Menor valor obtido	370	10.4	74075	15493	10496
Nº Soluções avaliadas	100	180	1091	1636	1426
Tempo execução	1,2 s	5,3 s	11,95 s	16,52 s	36,70 s
Valor ótimo de referência	370	10.4	72041	10574	5598
Diferença entre o ótimo e o melhor obtido neste estudo	0	0	+2034	+4919	+4898
Desvio do valor ótimo	0%	0%	2,83%	46,5%	87,49%

Tabela 6.4 - Resultados da aplicação do algoritmo aos problemas de sequenciamento em máquina única

Atentando agora na tabela 6.4, e na comparação dos valores das melhores soluções obtidas com os valores de referência (o valor ótimo para os problemas WT40 e WT50, e o melhor valor conhecido para o problema WT100, que foram obtidos a partir de uma base de dados da Universidade de Brunel em West London), conclui-se que este algoritmo encontra de facto soluções que podem ser consideradas boas, face à proximidade que existe entre os melhores valores obtidos e os valores ótimos conhecidos para os problemas. No problema WT40, por exemplo, a variação entre o valor ótimo do problema e o valor da melhor solução obtida com este algoritmo foi de 2,83%, que tendo em conta o número de soluções testadas,1091, revela a extrema eficiência deste algoritmo, em especial se se tiver em conta que para encontrar o ótimo global deste problema ter-se-iam que avaliar 40! soluções diferentes o que exigiria um esforço computacional imensamente superior ao despendido na

utilização deste algoritmo e um tempo muito maior do que os 11,95 segundos que este algoritmo necessitou para alcançar esta solução.

Para os problemas WT50 e WT100 o desvio da melhor solução obtida para a melhor solução conhecida é maior, e assume quase uma relação de proporcionalidade direta com o número de tarefas do problema, pois, como já foi explicado anteriormente para a variação do desvio padrão, a percentagem de soluções testadas face ao número de soluções possíveis é menor quanto maior for o número de tarefas a executar. Basta reparar que o número de soluções testadas para encontrar a melhor solução nos problemas WT40, WT50 e WT100 foi 1091, 1636 e 1426, respetivamente e o número de soluções possíveis para o WT100 é bastantes maior que para o WT40 e para o WT50.

Como se pode observar através dá análise do Gráfico 6.1 as soluções convergem para um ótimo local. A linha azul representa todas as soluções avaliadas pelo *Simulated Annealing*, enquanto que a linha vermelha representa a melhor solução obtida em cada momento da procura. O Algoritmo começa por encontrar soluções consideradas más, na ordem dos 200000, depois o valor das soluções encontradas vem descendo e vêm sendo aceites soluções diferentes, ou melhores que a atual ou piores que a atual mas aceites através da probabilidade de aceitação, que é decrescente no tempo, o que faz com que a variabilidade dos valores das soluções também seja decrescente no tempo. A procura converge para um ótimo local, onde a variabilidade das soluções acaba por estagnar e onde termina a procura pois o critério de paragem é atingido, no caso 250 corridas sem serem geradas soluções melhores que a atual.

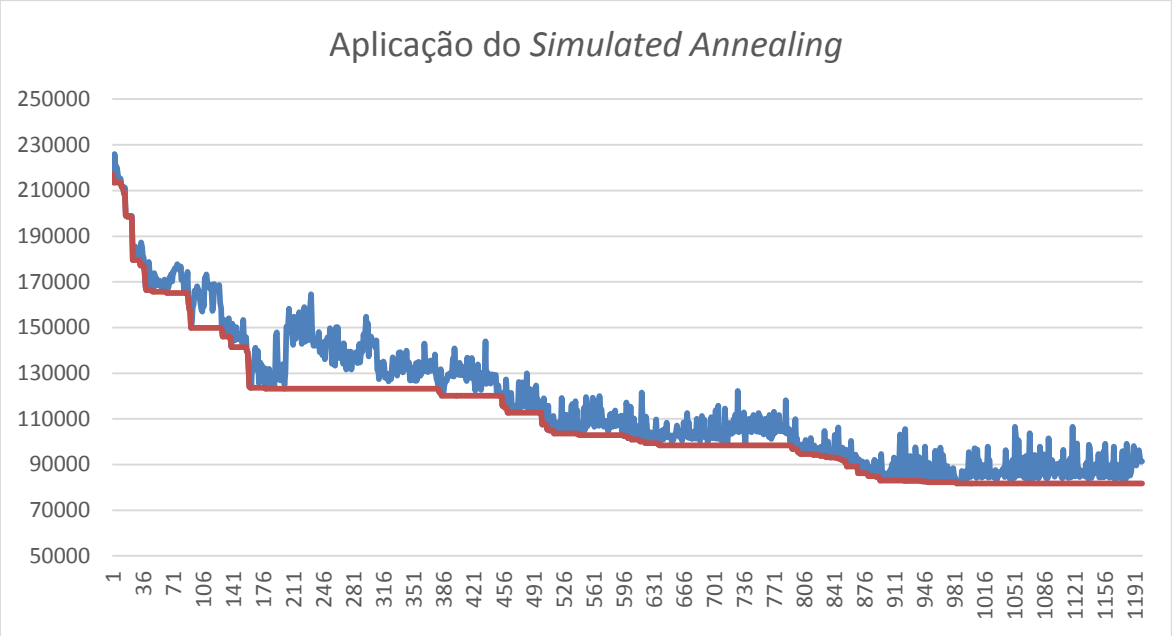

Figura 6.1 - Gráfico da pesquisa recorrendo ao *Simulated Annealing* para o problema WT40

Como foi discutido anteriormente se se atentar aos valores da percentagem de desvio da melhor solução obtida em relação ao valor ótimo/melhor valor conhecido dos problemas (Tabela 6.4), rapidamente se percebe que este é crescente com o número de tarefas que o problema engloba. Este facto não torna este procedimento menos válido, pois apesar do elevado desvio obtido, por exemplo, no problema WT100 (87,49%), as soluções não deixam de poder ser consideradas "boas soluções" para o problema, pois o desvio aumenta mas o número de soluções possíveis também é muito maior, e o tempo de processamento do programa continua a ser muito baixo, cerca de 36,76 segundos. O objetivo deste algoritmo passa por encontrar soluções melhores que as iniciais e se se olhar para o gráfico 6.2, que representa a variação das soluções geradas e das melhores soluções obtidas com o decorrer do processo iterativo, constatamos que a solução final encontrada, 10496, é bastante melhor que a solução com que o processo foi iniciado, aproximadamente 215000. Deve-se ter ainda em conta o número de soluções que foram testadas que é francamente inferior ao número total de soluções possíveis, representa menos de 1 % do total das soluções possíveis, o que mais uma vez realça a capacidade deste algoritmo para convergir rapidamente para soluções razoáveis/boas. A dificuldade para calcular um bom resultado para este problema é refletido pelo facto de não existir ainda uma solução, considerada ótima, para este problema, apesar estar disponível para resolução desde 2004.

No objetivo inicial deste estudo não se tinha em conta a influência da solução inicial no resultado final obtido, no entanto durante a realização dos testes computacionais foi percetível que a soluções iniciais "melhores" correspondem normalmente melhores soluções "finais". Caso a procura parta de uma solução viável próxima da solução ótima a probabilidade de encontrar a solução ótima aumenta bastante. Dai que tenha concluído de uma forma lógica que caso o algoritmo de "*Simulated Anealling*" seja repetido várias vezes consecutivas, iniciando a segunda corrida com o valor final da primeira a probabilidade de se alcançar melhores soluções aumenta de forma significativa. Esta teoria pode ser confirmada através da observação do gráfico 6.2, que espelha a aplicação desta teoria (foram efetuadas várias corridas do algoritmo consecutivas) ao problema WT50. A procura inicia em soluções com valores menores resultantes da corrida anterior, mas com uma probabilidade de aceitação elevada, o que permite explorar de uma forma mais ampla a vizinhança da melhor solução da corrida anterior, e originar frequentemente melhores soluções.

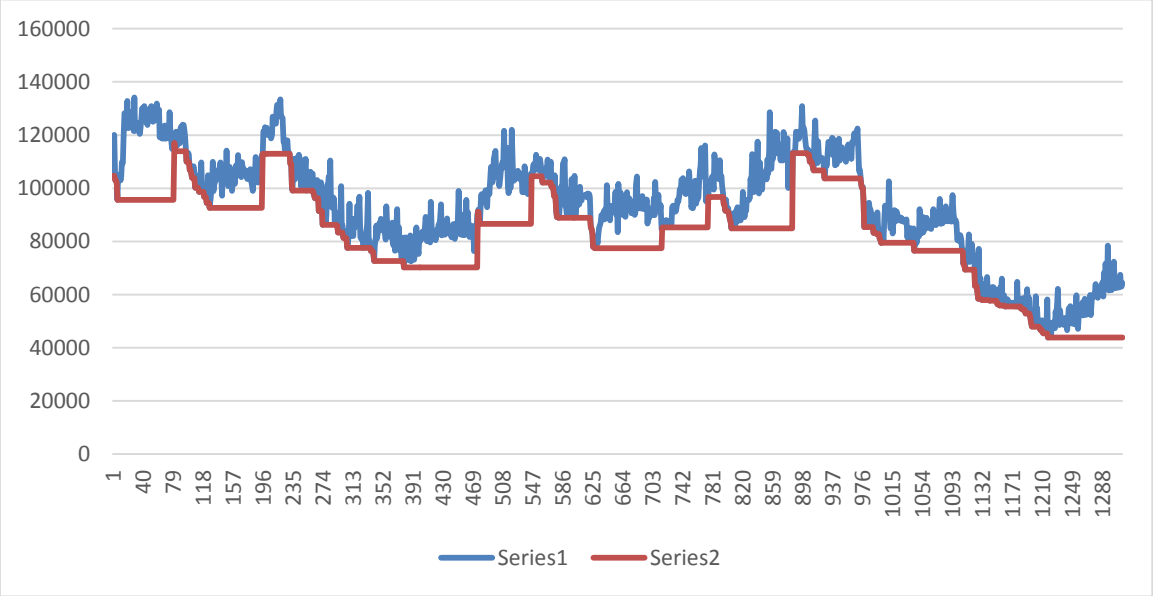

Figura 6.2 - Gráfico da aplicação sucessiva do algoritmo *Simulated Annealig* ao problema WT50

Outro facto de que me apercebi durante a elaboração deste trabalho foi que o critério de paragem poderia estar a "limitar", por vezes, o acesso a soluções ainda melhores que as tidas como "boas soluções" pelo algoritmo. Assim, optei por retirar o critério de paragem e o algoritmo passou a ser executado até que a temperatura mínima fosse atingida. O que aconteceu foi que por exemplo, no algoritmo original, para um critério de paragem de 250

iterações, depois de 250 iterações em que a melhor solução do problema não fosse melhorada o programa terminava. Com esta variação no algoritmo passadas essas 250 iterações a pesquisa prossegue até que a temperatura mínima seja alcançada. A probabilidade de aceitação depois de 250 iterações sem melhoria é muito baixa (aproximadamente 0) e por isso a busca cinge-se às soluções próximas da melhor solução encontrada (a solução que não foi melhorada nas ultimas 250 iterações), mas estas são exploradas de forma mais aprofundada. Nas corridas efetuadas a maioria das melhorias nas melhores soluções obtidas não foram dignas de grande relevo, no entanto acabou por ser obtida uma melhor solução para o problema WT50 com esta variação no algoritmo, à custa claro de um maior esforço computacional e de mais tempo de execução do programa, não deixando no entanto de ser um tempo baixo (cerca de 27,75s até terminar a procura) quando comparado com o necessário para analisar todas as soluções possíveis para este problema (50!). O gráfico 5.3 demonstra esta situação, com o algoritmo original a procura teria terminado por volta da iteração nº 2045, no entanto, nesta variação do algoritmo a procura continuou até à iteração nº2401 e acabou por encontrar um valor menor que o que teria sido dado como ótimo local no algoritmo original (iteração 2045).

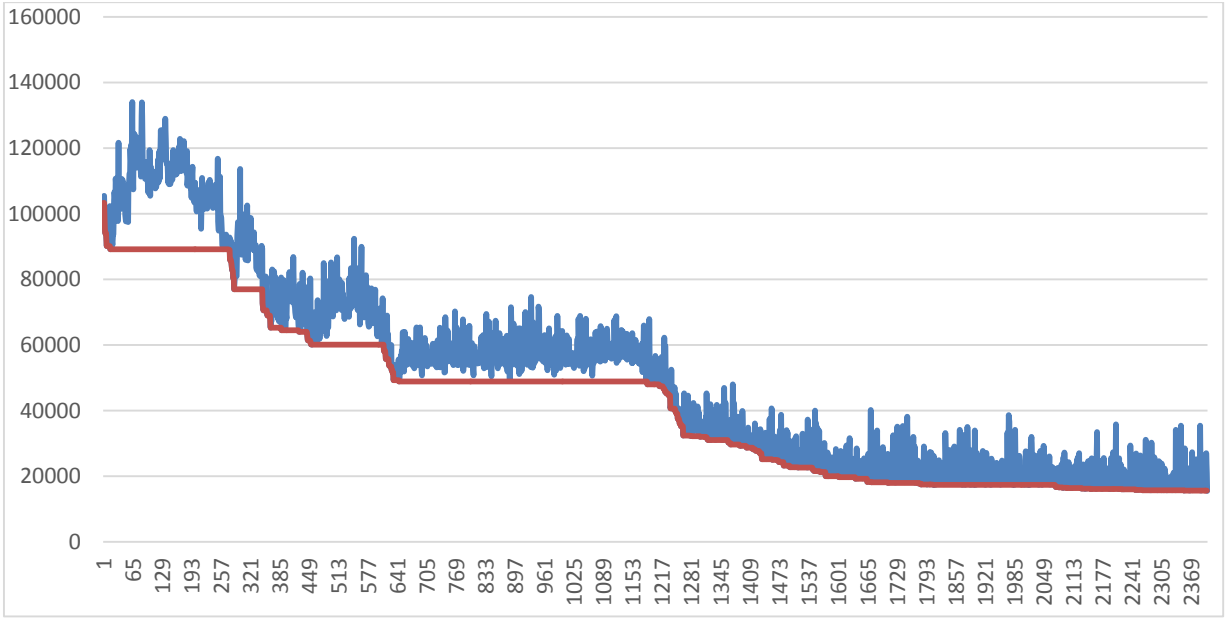

Figura 6.3 - Gráfico da aplicação da variação do algoritmo *Simulated Annealing* (sem critério de paragem) ao problema WT50

7. CONCLUSÃO

Como objetivo central deste trabalho pretendia-se demonstrar o potencial da utilização de algoritmos baseados em meta-heurísticas, mais propriamente baseados no *Simulated Annealing*, na resolução de problemas comuns no planeamento da produção. Depois de desenvolvidos, implementados computacionalmente e testados em dois tipos de problemas afetos ao planeamento da produção, estes algoritmos provaram que podem ser bastante úteis na resolução deste tipo de problemas pois conduzem, na maioria das vezes, a soluções satisfatórias de uma forma rápida e eficiente.

Concluiu-se ainda que, por serem algoritmos baseados na aleatoriedade, é aconselhável aplicar o algoritmo mais que uma vez a cada problema, de forma a evitar assumir como soluções finais soluções que não são tão boas quanto o algoritmo pode encontrar e que são decorrentes de convergências prematura do algoritmo. Seguindo esta linha de raciocínio constatou-se que a repetição sucessiva do algoritmo, começando a segunda corrida onde terminou a primeira e assim sucessivamente, e a abolição do critério de paragem podem, por vezes ser boas hipóteses para encontrar melhores soluções.

7.1. Desenvolvimentos futuros

Como foi discutido neste trabalho, os resultados obtidos neste trabalho demonstram que pode haver interesse em aprofundar a investigação tanto na utilização sucessiva e de forma repetida do algoritmo *Simulated Annealing*, como na utilização deste algoritmo sem definir um critério de paragem, guardando sempre a melhor solução obtida, tendo tanto numa proposta como na outra, sempre em conta o tempo disponível.

A introdução de uma heurística na definição da solução inicial baseada em regras básicas de prioridade como o SPT ou o EDD também pode ser interessante no sentido

em que pode melhorar o valor da solução inicial aquando do início da aplicação do algoritmo de *Simulated Annealing*.

Por outro lado também seria interessante expandir a utilização do algoritmo *Simulated Annealing* a outras funções objetivo que incluam, por exemplo, o custo associado a cada atraso, ou os tempos de *setup*. Este tipo de alterações seriam bastante fáceis de introduzir caso se tenha como base o processo utilizado no desenvolvimento deste trabalho, e seriam bastante uteis por exemplo num sistema de apoio à tomada de decisão.

8. REFERÊNCIAS BIBLIOGRÁFICAS

Antunes, C. H. (2013). *Aulas de Complementos de Investigação Operacional.* Coimbra.

Antunes, C. H., & Maria João Alves, J. C. (s.d.). *Tomada de Decisão em Ambiente Multiobjectivo.* Coimbra: Universidade de Coimbra.

Arenales, M., Armentano, V., Morabito, R., & Yanasse, H. (2006). *Pesquisa Operacional para Cursos de Engenharia. 1a Edição.* Rio de Janeiro: Elsevier.

Arenales, M., Armentano, V., Morabito, R., & Yanasse, H. (2007). *Pesquisa operacional para cursos de engenharia.* Campus.

Ashakov. (2013). *http://ashakhov.wordpress.com/2011/01/27/simulated-annealing.*

Baker, K. (1974). *Introduction to Sequencing and Scheduling.* Wiley, New York.

Baker, K. R. (2009). *Principles of Sequencing and Scheduling.* Wiley.

Batalha, M. O. (2008). *Introdução a engenharia de produção.* Rio de Janeiro: Elsevier.

Beasley, J. E. (2004). *Base de dados da Universidade de Brunel.* West London: http://people.brunel.ac.uk/~mastjjb/.

Brucker, P. .. (2004). *Scheduling Algorithms 4th edition.* Springer.

Casse, G. L. (2007). *A aplicação de simulação-otimização para definição do mix ótimo de produção de uma indústria metalo-mecânica. XXVII Encontro Nacional de Engenharia de Produção.* Foz do Iguaçu.

Easy, E. (2014). *Excel Tutorial on the net/VBA em http://www.excel-easy.com/vba.html.*

French, S. (1982). *Sequencing and Scheduling – An Introduction to Mathematics o the Job Shop.* New York: Wiley.

Goldbarg, M. C., & Luna, H. P. (2005). *Otimização combinatória e programação linear – Modelos e algoritmos. 2°ed.* Rio de Janeiro: Elsevier.

Izquierdo, V. B. (2000). Uma proposta de especificação formal e fundamentação teórica para simulated annealing. *Universidade Federal do Rio Grande do Sul.* Porto Alegre.

Júnior, A. d. (Março de 2007). Problema de Seqüenciamento em uma Máquina com Penalidades por Antecipação e Atraso: Modelagem e.

Kirkpatrick, S., Gellt, D. C., & Vecchi, M. P. (1983). *Optimization by Simulated Annealing.* Science.

Lawer, E. (1983). *Recem Results in Theory off Machine Scheduling, Mathematical Programming: The State of the Ar.* Springer Verlag: A. Bachen et al,.

M., P. (1998). *Computer simulation in management science (4th ed).* Chichester: John Wiley & Sons Ltd.

Madureira, A. M., & Sousa, J. P. (1996). *Aplicação de Meta-Heurísticas a problemas de Escalonamento de uma única Máquina.*

Mortonn, ,. T. (1993). *Heuristic Scheduliing Systems* . New York: Wiley.

Nurmela, K. (1993). Constructing combinatorial designs by local search. MSc thesis, Research report A27. *Digital Systems Laboratory.* Helsinki University of Technology,: Espoo.

Panneerselvam Senthilkumar, S. N. (2010). *Simulated Annealing Algorithm to Minimize Makespan in Single Machine Scheduling Problem with Uniform Parallel Machines.* Vellore, India.

Pinedo, M. (2002). *Scheduling: Theory, Algoritms, and Systems, second ed.* Prentice Hall.
Pozzo, A. (25/09/2010.). *Apresentação: Metaheuristicas.* Disponível em <http://www.inf.ufpr.br/aurora/disciplinas/topicosia2/aulas/aula6.pptx.
Silva, C. (2013). *Aulas da disciplina de Gestão da Produção.* Coimbra.
Silva, E. M., & al., e. (1998). *Pesquisa operacional para os cursos de economia, administração e ciências contábeis.* São Paulo: Atlas.
Varela,M., & Ribeiro,R.(Dez (2001)).Revista de Investigação Operacional, 21 - In Portuguese

ANEXO 1

```
acção = Int((99 - 0 + 1) * Rnd + 1)


If acção < 50 Then



    Sheet2.Range("E1:E20").Clear
    Sheet2.Range("F1:F20").Clear

    For i = 1 To UBound(Seq1)
    Sheet2.Range("E" & i + 13) = Seq1(i)
    Next i

    For i = 1 To UBound(Seq2)
    Sheet2.Range("F" & i + 13) = Seq2(i)
    Next i

    size1 = WorksheetFunction.CountA(Sheet2.Columns(5))
    size2 = WorksheetFunction.CountA(Sheet2.Columns(6))

            trf1 = Int((size11 - 1 + 1) * Rnd + 1)
            trf2 = Int((size12 - 1 + 1) * Rnd + 1)


            Dim tmp

            tmp = Seq1(trf1)
            Seq1(trf1) = Seq2(trf2)
            Seq2(trf2) = tmp

           For i = 1 To size1
           Sheet2.Range("E" & i + 13) = Seq1(i)
           Next


          For i = 1 To size2
          Sheet2.Range("F" & i + 13) = Seq2(i)
          Next

          Application.ScreenUpdating = True

         ElseIf acção >= 50 Then


                    ReDim Preserve Seq1(UBound(Seq1))
                    ReDim Preserve Seq2(UBound(Seq2))
```

```
                Sheet2.Range("E1:E20").Clear
                Sheet2.Range("F1:F20").Clear

                For i = 1 To UBound(Seq1)
                Sheet2.Range("E" & i + 13) = Seq1(i)
                Next i

                For i = 1 To UBound(Seq2)
                Sheet2.Range("F" & i + 13) = Seq2(i)
                Next i



                        size1 = WorksheetFunction.CountA(Sheet1.Columns(5))
                        size2 = WorksheetFunction.CountA(Sheet1.Columns(6))

                        Sheet2.Range("A5") = size1
                        Sheet2.Range("A6") = size2

        If quantos <> 2 Then

                    máquina = Int((2 - 1 + 1) * Rnd + 1)

                    bT = bT + 1


                    Sheets("Sheet2").Cells(bT, 11) = máquina

                    Sheet2.Range("A1") = máquina

                    t = t + 1

           If máquina = 1 Then

                    size1 = WorksheetFunction.CountA(Sheet2.Columns(5))
                    size2 = WorksheetFunction.CountA(Sheet2.Columns(6))

                    trf1 = Int((size1 - 1 + 1) * Rnd + 1)
                    tarefatroca = Seq1(trf1)


                    Sheet2.Range("E1:E20").Clear
                    Sheet2.Range("F1:F20").Clear

                    Seq1(trf1) = Seq1(UBound(Seq1))
                    ReDim Preserve Seq1(UBound(Seq1) - 1)
```

```
            ReDim Preserve Seq2(size2 + 1)

            Seq2(size2 + 1) = tarefatroca

            For i = 1 To size2 + 1
            Sheet2.Range("F" & i + 13) = Seq2(i)
            Next i

For i = 1 To size2 + 1
Sheet2.Range("F" & i + 13) = Seq2(i)
Next i


For i = 1 To size1 - 1
Sheet2.Range("E" & i + 13) = Seq1(i)
Next


Application.ScreenUpdating = True


ElseIf máquina = 2 Then

        size1 = WorksheetFunction.CountA(Sheet2.Columns(5))
        size2 = WorksheetFunction.CountA(Sheet2.Columns(6))

        trf2 = Int((size2 - 1 + 1) * Rnd + 1)

        trf2 = Int((size2 - 1 + 1) * Rnd + 1)
        tarefatroca = Seq2(trf2)


        Sheet2.Range("E1:E20").Clear
        Sheet2.Range("F1:F20").Clear

        ReDim Preserve Seq1(size1 + 1)

        Seq1(size1 + 1) = tarefatroca

        For i = 1 To size1 + 1
        Sheet2.Range("E" & i + 13) = Seq1(i)
        Next i

        Seq2(trf2) = Seq2(UBound(Seq2))
        ReDim Preserve Seq2(UBound(Seq2) - 1)
```

```
        For i = 1 To size2 - 1
        Sheet2.Range("F" & i + 13) = Seq2(i)
        Next

        Application.ScreenUpdating = True

    End If


         ElseIf size11 > size12 Then

                    trf1 = Int((size1 - 1 + 1) * Rnd + 1)
                    tarefatroca = Seq1(trf1)


                    Sheet2.Range("E1:E20").Clear
                    Sheet2.Range("F1:F20").Clear

                    Seq1(trf1) = Seq1(UBound(Seq1))
                    ReDim Preserve Seq1(UBound(Seq1) - 1)

                    ReDim Preserve Seq2(size2 + 1)

                    Seq2(size2 + 1) = tarefatroca

                    For i = 1 To size2 + 1
                    Sheet2.Range("F" & i + 13) = Seq2(i)
                    Next i


                    For i = 1 To size1 - 1
                    Sheet2.Range("E" & i + 13) = Seq1(i)
                    Next


                    Application.ScreenUpdating = True
```

```
        ElseIf size12 > size11 Then

                size1 = WorksheetFunction.CountA(Sheet2.Columns(5))
                size2 = WorksheetFunction.CountA(Sheet2.Columns(6))

                trf2 = Int((size2 - 1 + 1) * Rnd + 1)

                trf2 = Int((size2 - 1 + 1) * Rnd + 1)
                tarefatroca = Seq2(trf2)


                Sheet2.Range("E1:E20").Clear
                Sheet2.Range("F1:F20").Clear

                ReDim Preserve Seq1(size1 + 1)

                Seq1(size1 + 1) = tarefatroca

                For i = 1 To size1 + 1
                Sheet2.Range("E" & i + 13) = Seq1(i)
                Next i

                Seq2(trf2) = Seq2(UBound(Seq2))
                ReDim Preserve Seq2(UBound(Seq2) - 1)


                For i = 1 To size2 - 1
                Sheet2.Range("F" & i + 13) = Seq2(i)
                Next

                Application.ScreenUpdating = True

            End If

        End If
```