Faculdade de Engenharia da Universidade do Porto

Programa de optimização da procura horária em diagramas de patamares

Nuno Miguel Soares da Fonseca

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Junho de 2009

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2008/2009

A Dissertação intitulada

"Programa de optimização da procura horária em diagrama de patamares"

foi aprovada em provas realizadas em 20/Julho/2009

o júri

Presidente Professor Doutor José Rui da Rocha Pinto Ferreira
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Carlos Manuel Borralho Machado Ferreira
Professor Adjunto do Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia de Coimbra

Professor Doutor João Paulo Tomé Saraiva
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto
da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor – Nuno Miguel Soares da Fonseca

Nuno Miguel Soares da Fonseca

Faculdade de Engenharia da Universidade do Porto

Resumo

A procura de energia cada mais acentuada exige o desenvolvimento de estruturas e de modelos mais complexos capazes de acompanhar esta evolução. Este desenvolvimento traduz-se muitas vezes na criação ou melhoramento de aplicações computacionais.

No âmbito de previsões de carga, análises de ponta e base, estudos de possíveis desfasamentos de preços de electricidade em relação à potência consumida e análises de evolução das cargas e de preços, surge a necessidade de desenvolver diversos modelos e aplicações computacionais. Espera-se que sejam capazes de dar resposta às novas solicitações, interagindo com modelos já existentes. Num ambiente de consumo de electricidade sempre em constante mudança são necessários estudos de planeamento capazes de prever as necessidades futuras e de tomar decisões baseadas nessas previsões.

Neste âmbito foi implementada uma ferramenta versátil que é capaz de dar resposta a algumas destas solicitações. Esta ferramenta realiza a optimização do ajuste de diagramas de patamares a uma série de valores, tipicamente diagramas de energia eléctrica consumida ou de preços de energia eléctrica. A razão da utilização destes diagramas prende-se com a necessidade da interacção com outros modelos, por exemplo com o modelo VALORAGUA que será alvo de uma breve descrição.

O problema a resolver tem carácter combinatório, tendo sido usada a técnica *Simulated Annealing*, uma metaheurística muito usada neste tipo de problemas e que tipicamente alcança bons resultados. O seu sucesso reside principalmente na forma como consegue escapar aos óptimos locais.

Esta tese de dissertação surgiu no âmbito de um tema proposto pela EDP Produção, cujos técnicos acompanharam e aprovaram o seu desenvolvimento e os resultados obtidos.

Este trabalho inclui um capítulo que descreve as principais metaheurísticas existentes e um outro que enuncia trabalhos na área que utilizam a técnica *Simulated Annealing*. Em seguida, é referida a necessidade e funcionalidade da ferramenta implementada finalizando-se com a análise de resultados obtidos.

Abstract

With the demand for electricity increasingly pronounced, it is necessary to develop structures and more complex models capable of monitoring these developments. This development is reflected many times insure establishment or enhancement of computer applications.

Under load forecasts, analysis of peak base load, studies of possible differences in prices of electricity on power consumption and analysis of trends of demand prices, there is a need to develop new models and computational applications. It is expected that they are able to respond to new demands, interacting with already existing models. In an environment of electricity consumption changing, it is necessary to conduct planning studies able to predict future needs and make decisions based on this.

In this follow-up, it was implemented a versatile tool that is able to respond to some of these requests. This tool optimizes the adjustment of diagrams of steps to a series of values, typical energy consumption or prices of electricity. The reason of using diagrams is the need of interaction with other existing models, as the model VALORAGUA that is breefly described in this text.

The developed optimization problem hás combinatorial nature motivating the use of Simulated Annealing technique, a very used metaheuristic in this type of problems, which typically achieves good results. Its success mainly lies in the capacity to escape from local optimum.

This thesis emerged from a proposal submitted by EDP Produção. Engineers from this company approved and followed its development and the obtained results.

In this work there is a chapter that describes the main existing metaheuristics and another that addresses some works in the field using the Simulated Annealing technique. There is a chapter that describes the need and functionality of the implemented tool and another dedicated to the analysis of results that were obtained.

Agradecimentos

- Aos meus pais, pelo carinho, apoio, pela confiança demonstrada e por me darem condições para que um dia fosse Engenheiro;

- À Diana, pelo amor, presença constante e união nas horas difíceis;

- Aos meus irmãos, pelo apoio e por serem um exemplo a seguir;

- Ao Cristiano e ao Pedro, verdadeiros companheiros e amigos durante estes meses e a todos os amigos e colegas que frequentaram esse grande edifício que é o J;

- Aos meus amigos já Engenheiros, pelo convívio e pelos conselhos fornecidos;

- Aos amigos da residência Jayme Rios de Souza, pelo companheirismo e amizade ao longo de todos estes anos;

- Ao professor Tomé Saraiva, pela orientação, disponibilidade e oportunidade de realizar este trabalho;

- À EDP Produção pela oportunidade oferecida e em particular aos Engenheiros Virgílio Mendes e José Carlos Sousa pelo esclarecimento de dúvidas e pela simpatia sempre demonstrada;

- Ao Engenheiro Jorge Pereira, pelo acolhimento e esclarecimento de dúvidas fulcrais no desenvolvimento deste trabalho.

- Ao Bruno Aleixo, ao homem do Buçaco, ao Nelso, ao Busto, ao Ribeiro e a todos os amigos do café do Aires

"O único lugar onde sucesso vem antes do trabalho é no dicionário"

Albert Einstein

"Nada está feito enquanto resta alguma coisa para fazer"

Romain Rolland

"Passar muito tempo a estudar é preguiça"

Francis Bacon

Índice

Lista de figuras

Lista de tabelas

Capítulo 1

Introdução

1.1 Aspectos Gerais

Com a crescente complexidade da rede energética nacional e o desenvolvimento de mercados transaccionais, surge a necessidade de desenvolver e utilizar modelos igualmente complexos e que respondam adequadamente às exigências da rede. O planeamento do sistema eléctrico é por isso muito importante a médio e a longo prazo, uma vez que com um planeamento adequado se atenuarão os custos associados e conseguir-se-á uma actuação mais rápida e eficaz.

No âmbito do planeamento do sistema eléctrico, existem várias abordagens que têm de ser feitas incluindo a previsão de cargas, análise de futuras expansões e a comparação de preços de electricidade em dois países. No que se refere, por exemplo, à previsão de cargas, é útil possuir uma análise de evolução de cargas. Para tomar determinadas decisões, como por exemplo se é rentável realizar bombagem num determinado período, é útil analisar relações entre a ponta e a base do diagrama de cargas. No entanto, estes estudos e análises devem ser realizados considerando os modelos que já existem.

Neste sentido, torna-se vantajoso criar modelos matemáticos e computacionais versáteis e flexíveis que encontrem uma solução para estas necessidades, não descurando a interacção exigida com modelos já existentes.

Tendo em conta todos os aspectos referidos, a EDP Produção propôs o desenvolvimento de uma aplicação computacional que fosse útil para os seus estudos de planeamento e cujo objectivo consiste em optimizar diagramas de patamares ajustando-os em função dos valores em estudo.

1.2 Motivação e objectivos

O objectivo deste trabalho foi o de desenvolver uma aplicação computacional capaz de receber valores horários relativos a um período de tempo previamente especificado, e criasse um diagrama de cinco patamares decrescentes que se aproximasse com o mínimo de erro a aos valores dados. Assim sendo, foi implementado um algoritmo em *Visual Basic Apllications* (VBA) que é capaz de criar até cinco diagramas de patamares em simultâneo que acompanham valores de períodos de tempo distintos minimizando o erro.

O valor do erro foi calculado tendo por base a área que é formada entre os valores inseridos, tipicamente valores de potência consumida ou preços de electricidade, e o topo dos patamares. Este valor de erro constitui o valor da função de avaliação do *Simulated Annealing*. As soluções encontradas distinguem-se pelas larguras e alturas de cada patamar, e é através da sua mudança de forma iterativa que são identificadas as melhores soluções. Quando são feitas optimizações de diagramas simultâneos é imposta uma restrição relacionada com o facto de a largura de um determinado patamar dever ter o mesmo valor em todos os diagramas.

Como se trata de um problema altamente combinatório, foi decidido usar uma metaheurística no processo de pesquisa de soluções. O *Simulated Annealing* foi a técnica escolhida e adaptada ao problema em questão, por se tratar de uma técnica de fácil implementação computacional e por conduzir a bons resultados como comprovam os resultados obtidos por diversos trabalhos.

A ferramenta desenvolvida deverá interagir com o modelo VALORAGUA que é um modelo de planeamento do sistema eléctrico. As características do problema, tais como a necessidade de criar diagramas em patamares e outras restrições, devem-se essencialmente ao interesse em realizar esta interacção. A aplicação computacional serve então para ajudar no processo do planeamento eléctrico, nomeadamente na realização de análise de evolução de energia eléctrica consumida, evolução de preços de electricidade durante um período de tempo, análise da relação ponta–base úteis para verificar, por exemplo, se a bombagem é rentável, entre outras análises possíveis.

1.3 Estrutura

Este trabalho está dividido em seis capítulos. O presente capítulo, Capítulo 1, consiste numa introdução genérica sobre o trabalho, assim como na definição da estrutura do documento.

O segundo capítulo resume os principais conceitos sobre algumas das mais importantes metaheurísticas existentes, dando especial enfoque ao *Simulated Annealing* por ser a metaheurística que é utilizada para resolver o problema de optimização que foi formulado.

O Capítulo 3 consiste numa pesquisa bibliográfica sobre trabalhos existentes na área que usam o *Simulated Annealing* como abordagem a esse mesmo problema. São relatados quatro problemas distintos da área de Energia.

O quarto capítulo explica a necessidade, o desenvolvimento, as considerações adoptadas da ferramenta criada. É ilustrado o ambiente de trabalho da ferramenta e as suas capacidades e flexibilidade são também abordadas. São apresentadas ainda todas as expressões matemáticas inerentes ao problema e um fluxograma que descreve todo o algoritmo desenvolvido.

O quinto capítulo demonstra a aplicação real da ferramenta usando para isso quatro casos de estudo com dados de carga e preços fornecidos pela EDP Produção. A partir deles foram realizadas diversas simulações para posterior análise e discussão.

O último capítulo, Capítulo 6, consiste no resumo dos principais passos da realização deste trabalho, enunciando-se igualmente as conclusões que daí foram obtidas.

Capítulo 2

Metaheurísticas

2.1 Ideias Gerais

Antes de qualquer introdução teórica sobre as metaheurísticas e, em geral, sobre métodos de optimização, é fundamental resumir alguns conceitos essenciais de forma a situar o contexto dos parágrafos seguintes.

Definição 1. (Optimização) – Um problema de optimização está normalmente associado ao par (S,f), onde S≠0 representa o espaço de soluções – ou espaço de pesquisa – do problema, enquanto f é um critério de qualidade conhecido como função objectivo, definida como:

$$f:S \rightarrow \Re \tag{2.1}$$

Assim, resolver um problema de optimização consiste em encontrar um conjunto de valores para as variáveis de decisão tal que a solução representada por esses valores $i^* \in S$ satisfaz a seguinte desigualdade, no caso de se tratar de um problema de minimização:

$$f(i^*) \leq f(i), \ \forall \ i \in S \tag{2.2}$$

Assumindo que a maximização ou minimização não restringe a generalidade dos resultados, podemos estabelecer uma equivalência entre problemas de maximização e minimização tal que:

$$\text{Max} \{ f(i) \mid i \in S\} \equiv \min \{-f(i) \mid i \in S\} \tag{2.3}$$

De acordo com o domínio de S, podemos definir uma optimização binária ($S \subseteq B$), optimização completa ($S \subseteq N$), optimização contínua ($S \subseteq \Re$), ou problemas mistos ($S \subseteq \{B \cup N \cup \Re\}$).

A definição de proximidade entre duas soluções do espaço de pesquisa é necessária para resolver um problema de optimização. Duas soluções estão próximas uma da outra se pertencerem à mesma vizinhança no espaço de pesquisa. A definição de vizinhança é indicada em seguida [1].

Definição 2 - (Vizinhança) – Seja (S, f) um problema de optimização. Uma estrutura de vizinhança em S pode ser definida como:

N:S $\rightarrow$ S, tal que para cada solução i $\in$ S um conjunto $S_i \subseteq$ S seja definido. Isto significa também que se i está na vizinhança de j então j está na vizinhança de i:j $\in S_i$ se i $\in S_j$.

Em geral, num problema de optimização complexo é frequente a função objectivo apresentar uma solução óptima que é um óptimo apenas numa determinada vizinhança, mas não apresentar uma solução óptima global quando se considera todo o espaço de pesquisa. Assim sendo, o método de pesquisa pode facilmente ficar preso num valor óptimo associado a uma vizinhança, levando assim ao conceito de óptimo local, apresentado a seguir [1].

Definição 3 - (Óptimo local) – Seja (S,f) um problema de optimização, e $S_{i'} \subseteq$ S uma vizinhança da solução i' $\in S_{i'}$, i' é um óptimo local se a seguinte desigualdade for satisfeita assumindo um problema de minimização.

$$f(i') \leq f(i), \ \forall i \in S_{i'} \tag{2.4}$$

Em problemas da vida real, somos normalmente forçados a lidar com restrições. Nesses casos, o espaço de soluções admissíveis é limitado de forma a satisfazer todas as restrições [1].

Existem muitas propostas de técnicas e de algoritmos na literatura, tanto exactas como aproximadas, para resolver problemas de optimização. Algoritmos exactos garantem a identificação da solução óptima para todos os casos finitos. Uma distinção mais eficaz pode ser observada na Figura 2.1. Geralmente, uma vez que os métodos exactos necessitam de elevados cálculos computacionais, alguns problemas não podem mesmo ser resolvidos por estes métodos. Por essa razão, nas últimas décadas tem aumentado o uso de técnicas aproximadas. Com estes métodos perdemos a garantia de encontrar o óptimo global (frequentemente, mas nem sempre). Porém, com frequência são encontradas boas soluções num tempo relativamente curto quando comparado com os métodos exactos [1].

Nas duas últimas décadas, um novo tipo de técnicas aproximadas tem emergido, consistindo em combinar diferentes métodos heurísticos básicos entre eles, em ambientes de alto nível, de forma a explorar o espaço de pesquisa eficientemente. Estes métodos são normalmente conhecidos como metaheurísticas. Há várias definições, mas no geral pode-se definir uma metaheurística como uma estratégia de alto nível que contém uma dada estrutura que planifica um conjunto de operações para explorar espaços de pesquisa complexos [1].

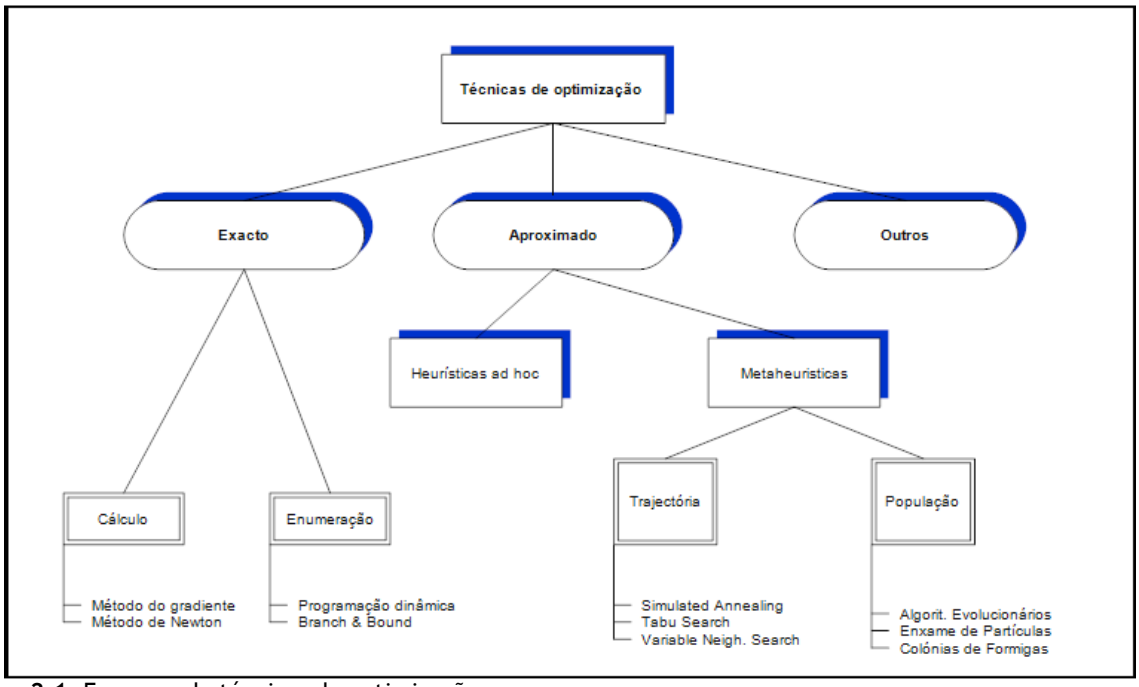

Figura 2.1- Esquema de técnicas de optimização.

Metaheurísticas, na sua definição original, são técnicas de solução que combinam uma interacção entre os procedimentos de melhoria local e maior nível de estratégias, para criar um processo capaz de escapar a óptimos locais, realizando uma pesquisa robusta do espaço de soluções. Ao longo do tempo, estes métodos têm vindo a incluir alguns procedimentos que empregam estratégias para superar a armadilha da optimalidade local no espaço de soluções complexo, especialmente os procedimentos que empregam uma ou mais estruturas de vizinhança como forma de definir mudanças admissíveis na transição de uma solução para outra, ou para construir ou eliminar soluções em processos de construção e destruição [2].

O grau de exploração das vizinhanças varia de acordo com o procedimento utilizado. No caso de certos procedimentos, tais como os algoritmos genéticos, as vizinhanças estão implicitamente (e de certa forma restritamente) definidas pela substituição de componentes de uma solução pelos de outra, variando a escolha em função de regras de troca popularmente conhecido como "*crossover*" [2].

Noutros métodos de base populacional, baseadas na noção de "*path relinking*", as estruturas de vizinhança são usadas na sua generalidade, incluindo vizinhanças construtivas e destrutivas, bem como para transição entre soluções [2].

Certas abordagens evolucionárias clássicas, ligadas à pesquisa local, também usam estruturas de vizinhança mais completas, embora para além disso incluam também processos combinatórios. Entretanto, abordagens do tipo "*single thread*", que não comprometem a manipulação de múltiplas soluções simultaneamente, executam uma ampla gama de processos combinatórios que não somente manipula diversas vizinhanças, mas incorpora numerosas estratégias desde aleatórias a determinísticas, dependendo de elementos tais como fase da pesquisa, ou no caso de métodos baseados em memória, da história do processo de solução [2].

Um número considerável de ferramentas e mecanismos que surgiram com a criação das metaheurísticas, provaram ser muito eficazes. É tanto assim que, nos últimos anos, as metaheurísticas têm sido o foco das atenções, sendo consideradas como uma forma adequada para resolver muitos problemas complexos, especialmente de natureza combinatória [2].

Enquanto as metaheurísticas não são capazes de garantir a optimalidade das soluções que encontram, os procedimentos exactos (que supostamente deveriam garantir essa optimalidade) têm-se mostrado incapazes de encontrar soluções cuja qualidade esteja próxima à obtida pelas metaheurísticas, principalmente em problemas reais caracterizados por elevados níveis de complexidade [2].

Estes resultados têm motivado investigação adicional bem como a aplicação e a melhoria de novas metodologias baseadas em metaheurísticas [2].

2.2 Principais Metaheurísticas existentes

Nos parágrafos que se seguem serão descritos os pormenores e curiosidades relativas às principais Metaheurísticas existentes. O estudo bibliográfico deste tópico incidiu sobre os seguintes métodos aproximados, deixando para última análise a técnica que foi utilizada neste projecto, *Simulated Annealing*:

- *Scatter Search* (SS) e *Path Relinking* (PR);
- *Tabu Search;*
- Algoritmos Genéticos;
- *Variable Neirghborhood Search* (VAS);
- *Guided Local Search* (GLS);
- *Greedy Randomized Adaptative Search Procedures* (GRASP);

- *Colónia* de formigas – *Ant Colony;*
- *Simulated Annealing.*

2.2.1 Scatter Search e Path Relinking

Scatter Search (SS) é uma técnica baseada na população que recentemente tem demonstrado resultados promissores para resolver problemas combinatórios e problemas de optimização não lineares. Baseado em formulações originalmente propostas na década de 60 que combinavam regras de decisão e restrições de problemas, SS utiliza estratégias para combinar vectores de soluções e tem provado ser eficiente em variados problemas [2].

Path Relinking (PR) tem sido sugerido como uma abordagem para diversificar estratégias no processo de pesquisa. A abordagem pode ser vista como um exemplo extremo (altamente focado) de uma estratégia que visa incorporar atributos de soluções de alta qualidade, através da criação de incentivos para favorecer esses atributos nos movimentos seleccionados [2].

2.2.1.1 Algoritmo *Scatter Search*

Scatter Search foi desenhado para operar com um conjunto de pontos, chamados de *pontos de referência,* que constituem boas soluções obtidas a partir de soluções anteriores. Esta abordagem gera sistematicamente combinações de *pontos referência* para criar novos pontos, sendo cada um dos pontos mapeado num ponto viável associado. Estas combinações constituem formas generalizadas de combinações lineares, acompanhadas por processos que impõem condições de viabilidade, incluindo aquelas de natureza discreta [2].

O procedimento indicado no parágrafo seguinte começa com a criação de uma referência inicial de um conjunto de soluções (*RefSet*). A diversificação do método de geração é usado para construir um conjunto alargado de diversas soluções ***P***. O tamanho de ***P*** (***PSize***) é tipicamente 10 vezes superior a *RefSet*. Inicialmente, a referência fixa *RefSet*, com *b* soluções distintas e diversificadas. As soluções em *RefSet* são ordenadas de acordo com a sua qualidade, ocupando a melhor solução a primeira posição na lista. A pesquisa é então iniciada atribuindo o valor VERDADEIRO à variável boleana *NovasSoluções*. No passo 3, *NovosSubConjuntos* é criado e *NovasSoluções* é trocado para FALSO. Para efeitos ilustrativos serão focados com mais atenção os subconjuntos de tamanho 2. Daí a cardinalidade dos *NovosSubConjuntos* corresponder à referência inicial dada por $(b^2-b)/2$, que é válida para todos os pares de soluções em *RefSet*. Os pares de *NovosSubConjuntos* são seleccionados um de cada vez numa ordem lexicográfica e o Método de Combinação de Soluções é aplicado

para gerar uma ou mais soluções no passo 5. Se uma solução recém-criada melhora uma solução pior em *RefSet*, a nova solução substitui a pior e *RefSet* é reordenado no passo 6. O valor de *NovasSoluções* é mudado para VERDADEIRO e o subconjunto *s* que tinha sido acabado de combinar é apagado de *NovosSubConjuntos* nos passos 7 e 8, respectivamente [2].

Algoritmo:

Começar com P=0. Usar o Método de Geração Diversificada para construir a solução ***x***. Se ***x*** $\notin$ P então adicionar ***x*** a P (i.e., $P=P\cup x$), senão, rejeitar ***x***. Repetir este passo até $|P| = P_{size}$. **Contruir** *RefSet* ={x^1,…,x^b} com *b* soluções diversas em *P*.

Avaliar as soluções em *RefSet* e ordena-las de acordo com a função objectivo de tal forma que x^1 é a melhor solução e x^b a pior. Colocar *NovasSoluções* = VERDADEIRO.

Enquanto (*NovasSoluções*) **fazer**

Gerar *NovosSubConjuntos*, que inclui todos os pares de soluções em *RefSet* que incluem pelo menos uma nova solução. Colocar *NovasSoluções* = FALSO.

Enquanto (NovoSubConjunto ≠ 0) **fazer**

Seleccionar o novo subconjunto seguinte, *s*, de *NovosSubConjunjos*

Aplicar o Método de Combinação de Soluções a *s*, de forma a obter uma ou mais novas soluções ***x***.

Se (***x*** não está em *RefSet e f(x)* < $f(x^b)$ **então**

Colocar x^b=***x*** e reordenar *RefSet*.

Colocar *NovasSoluções* = VERDADEIRO

Fim do se

Apagar s de NovosSubConjuntos

Fim do enquanto

Fim do enquanto

2.2.1.2 *Path Relinking*

Um dos principais objectivos em cada em qualquer método de pesquisa é criar um equilíbrio entre procura intensiva e procura diversificada. Alguns recursos que têm sido adicionados ao *Scatter Search*, foram também incluídos na técnica do *Path Relinking*. Esta abordagem gera novas soluções explorando trajectórias que unem soluções de alta qualidade, começando por uma dessas soluções, chamada de solução inicial, e gerando um caminho no espaço de vizinhança que a leva em direcção a outras soluções, chamadas de *guiding*

solutions. Isto é conseguido seleccionando movimentos que introduzem atributos contidos nas *guinding solutions* [2].

A abordagem é chamada *path relinking* por gerar um novo caminho (*path*) entre soluções previamente ligadas por uma série de movimentos durante a pesquisa, ou por gerar um caminho entre soluções previamente ligadas a outras soluções, mas não ligadas entre elas. A Figura 2.2 mostra dois caminhos hipotéticos (i.e. uma sequência de movimentos) que ligam a solução A à solução B, para mostrar religação do primeiro tipo [2].

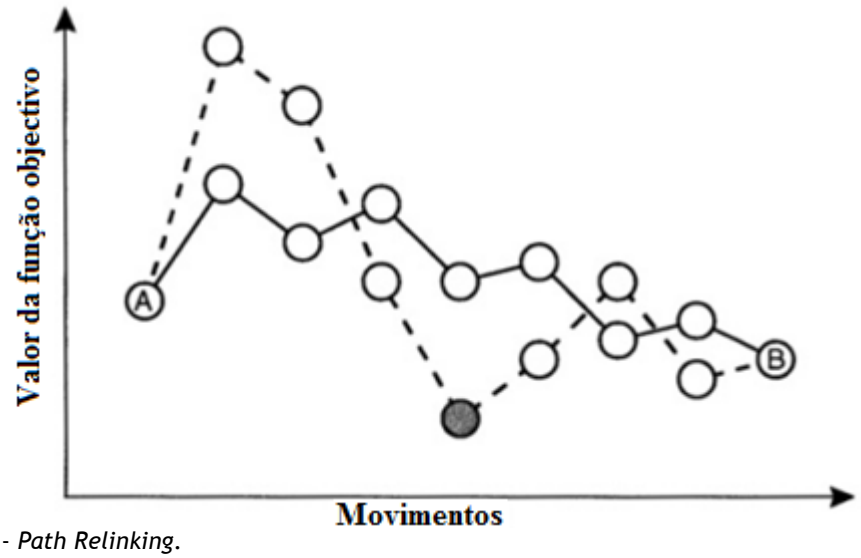

Figura 2.2 – *Path Relinking*.

A linha a cheio indica um caminho original produzido por uma operação "normal" de um procedimento que produz uma série de movimentos levando desde A até B, enquanto a linha tracejada representa o caminho de ligação (*path relinking*) [2].

As soluções são diferentes devido à selecção do movimento durante a operação normal não "saber" onde reside a solução B até a ter finalmente encontrado, mas simplesmente segue uma trajectória intermediada por passos que são determinados pela função de avaliação. Por exemplo, uma abordagem comummente usada consiste em seleccionar um movimento que minimiza (ou maximiza) o valor da função objectivo localmente. Durante o caminho de ligação, porém, o objectivo a atingir consiste em incorporar atributos da *guiding solution* enquanto ao mesmo tempo são gravados os valores da função objectivo [2].

O esforço para representar o processo num diagrama simples, tal como o anterior, cria no entanto algumas impressões enganosas. Em primeiro lugar, o caminho original (linha a cheio), que é mostrado para ser "ganancioso" relativamente à função objectivo, é susceptível de ser significantemente mais sinuoso ao longo de dimensões que não se é capaz de mostrar e, ao mesmo tempo, envolve mais passos significativos (soluções intervenientes) – um aspecto não retratado na figura acima. Segundo, o caminho ligado não é tão determinado pela atracção local mas, em vez disso, é influenciado pela incorporação de atributos da *guiding solution,*

abrindo a possibilidade de alcançar soluções melhores que não poderiam ser encontradas numa pesquisa "míope local". A Figura 2.2 mostra uma tal solução alcançada pelo caminho a tracejado. Para além disto, porém, o caminho ligado pode encontrar soluções que, não sendo melhores que a inicial ou que a *guiding solution*, fornecem pontos de acesso férteis para alcançar outras soluções, pouco melhores. Por esta razão, esta técnica pode ser útil para examinar as soluções vizinhas ao longo do caminho ligado, e acompanhar aquelas de alta qualidade que podem proporcionar um ponto de partida para uma pesquisa continuada [2].

2.2.2 *Tabu Search*

Evoluindo no seu trabalho prévio, Fred Glover propôs em 1986 uma nova abordagem à qual chamou *Tabu Search (TS)*, para permitir que os métodos *Local Search* (LS) superassem óptimos locais. O princípio básico do TS é seguir LS sempre que este encontre um óptimo local, sendo permitidos movimentos que não melhoram a função objectivo. O retorno cíclico a soluções pré visitadas é impedido usando memórias, chamadas de listas tabu, que recordam a história recente da pesquisa, uma ideia que está ligada a conceitos de Inteligência Artificial. É interessante notar que, no mesmo ano, Hansen propôs uma abordagem similar a que chamou *steepest ascent/mildest descent* [2].

Espaço de pesquisa e estrutura de vizinhança

Os dois principais elementos de qualquer heurística TS são a definição do espaço de pesquisa e a sua estrutura de vizinhança. O espaço de pesquisa de uma heurística TS é simplesmente o espaço de todas as soluções possíveis que podem ser consideradas (visitadas) durante a pesquisa. Associado à definição de espaço de pesquisa está a estrutura de vizinhança. Em cada iteração da TS, as transformações locais que podem ser aplicadas à solução actual, denotada de S, definem um conjunto de soluções vizinhas no espaço de pesquisa, denotado de N(S) (a vizinhança de S). Formalmente, N(S) é um subconjunto do espaço de pesquisa definido por:

N(S) = {soluções obtidas aplicando uma única transformação local a S}

Em geral, para um problema específico em análise, há muitas outras possibilidades (e até mais atractivas) de definir a vizinhança do que definições considerando o espaço de pesquisa. Isto provém do facto de haver várias estruturas de vizinhança aceitáveis para cada definição de espaço de pesquisa [2].

Tabus

Tabus são aqueles elementos distintivos da TS quando comparados com LS. Como já foi dito, os tabus são usados para prevenir movimentos cíclicos quando a pesquisa se afasta de óptimos locais através de movimentos que não melhoram a solução. É importante referir que quando esta situação ocorre, é necessário fazer alguma coisa para prevenir o processo de pesquisa de voltar atrás nos seus passos para a solução de onde veio. Isto é conseguido declarando movimentos tabu (impeditivos), que revertem o efeito dos recentes movimentos [2].

Os tabus são guardados numa memória de curto prazo da pesquisa (a lista dos tabus) e normalmente só uma pequena quantidade de informação é guardada. Num dado contexto, há várias possibilidades no que diz respeito à informação guardada. Uma pode gravar soluções completas, mas requer grande espaço de armazenamento e torna mais demorada a verificação se um dado movimento é tabu ou não sendo, por isso, raramente utilizada. A maior parte grava as últimas pequenas alterações na solução actual e proíbe transformações contrárias. Outras são baseadas nas características principais das próprias soluções ou dos movimentos [2].

Critérios de aspiração

Enquanto centro da TS, os tabus são por vezes demasiado poderosos: podem proibir movimentos atractivos, mesmo quando não há perigo de movimentos cíclicos, ou podem levar a uma estagnação do processo de pesquisa. É assim necessário usar algoritmos que cancelem ou mitiguem os tabus. Estes são chamados de critérios de aspiração. O critério de aspiração mais simples e mais usado (encontrado em quase todas as implementações) consiste em permitir um movimento, mesmo sendo tabu, se este resultar numa solução com um valor objectivo melhor que a melhor solução corrente (uma vez que a nova solução ainda não foi obviamente visitada). Critérios de aspiração muito mais complicados foram propostos mas são raramente usados. O aspecto principal nestes critérios é que se não for possível a ocorrência de movimentos cíclicos então os tabus podem ser ignorados [2].

Modelo geral e critério de paragem

Estamos agora em posição de indicar um modelo geral para a TS. Admitamos que estamos a minimizar uma função de um qualquer domínio e aplicamos a versão chamada “melhor melhoria” da TS, i.e., a versão na qual é escolhida em cada iteração o melhor movimento aceitável (esta é a versão mais comummente utilizada na TS).

Notação

S, a solução corrente

*S**, a melhor solução conhecida

*f**, valor de *S**

N(S), a vizinhança de *S*

Ñ(S), o subconjunto admissível de *N(S)*, ou seja, um não-tabu ou permitido por aspiração

Inicialização

Escolher (construir) uma solução inicial S_0

Colocar $S = S_0$, $f = f(S_0)$, $S^* = S_0$, $T = 0$

Pesquisar

Enquanto critério de paragem não for satisfeito fazer

Seleccionar *S* em argmin $[f(S')]; S' \in N'(S)$

Se $f(S) < f^*$, então colocar $f^* = f(S)$, $S^* = S$

Guardar tabu para movimento corrente em *T* (apagar anterior entrada se necessário)

Fim do enquanto [2]

No modelo indicado acima falta especificar o critério de paragem. Em teoria, a pesquisa poderia durar para sempre, a não ser que o valor óptimo do problema em análise fosse já conhecido. Na prática, obviamente, a pesquisa tem de parar em algum momento. Os critérios de paragem mais usados são:

- Fixar um número máximo de iterações que se admite realizar;
- Especificar um número de iterações ao fim das quais se não houver melhoria do valor da função objectivo o processo termina (o critério mais usado nas implementações);
- Quando o objectivo alcança um valor limiar pré especificado.

Em sistemas complexos, a pesquisa é normalmente interrompida depois de completar uma sequência de fases, sendo a duração de cada fase determinada por um dos critérios acima [2].

Conclusões da *Tabu Search*

Tabu Search é um algoritmo poderoso que tem sido aplicado com grande sucesso a imensos problemas combinatórios. Uma particularidade simpática da TS é que, tal como todas as abordagens baseadas em pesquisas locais, pode facilmente lidar com as restrições que aparecem nas aplicações da vida real [2].

Alguns defeitos deste método estão relacionados com o aparecimento de motivos ou de atractivos por consequência da exploração da vizinhança e com o inconveniente da propensão à descida muito rápida em direcção ao óptimo local. No que diz respeito ao primeiro caso, é

fundamental observar a Figura 2.3, onde os pontos A e B representam duas soluções possíveis obtidas durante a processo de pesquisa e V(A) e V(B) representam as suas vizinhanças [4].

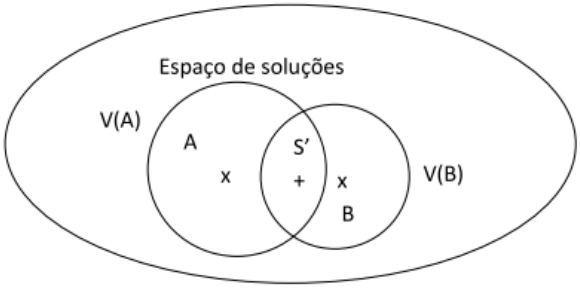

Figura 2.3 - Efeito funil sobre as vizinhanças [4].

Se as listas ou os critérios Tabu os permitirem, S' (solução corrente) será escolhida nos dois casos como a nova solução corrente. Segue-se uma sequência de soluções exploradas S_0=S', S_1, S_2, ..., S_n idêntica nos dois casos à qual é dado o nome de *motivo*. Esta sequência será mais ou menos longa segundo os diferentes estados da lista Tabu ou os mecanismos escolhidos, mas ela será prejudicial para este método. Com efeito, não somente se perderá tempo reavaliando várias vezes as mesmas soluções, como também não se ganhará conhecimento suplementar sobre o espaço de soluções [4].

Por outro lado, a descida agressiva corresponde também ela a um efeito nocivo da exploração exaustiva da vizinhança, visto que o método tem uma tendência para direccionar a trajectória de soluções em ordem a mínimos locais, de onde é difícil sair. Esta situação é ilustrada na Figura 2.4. A pesquisa prossegue desde S^0 (solução inicial) descendo até S* (solução óptima local). Em seguida, torna a subir para ser atraído para o óptimo local. Esta situação, onde a parte explorada do espaço das soluções oscila no círculo tracejado na figura, é também geradora de motivos onde o comprimento coincide com o da lista *Tabu*. Assim, resulta destes inconvenientes que o espaço das soluções é explorado de forma muito incompleta e insuficiente.

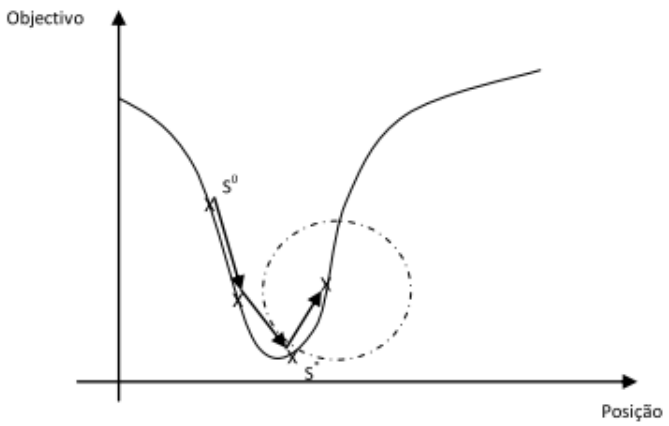

Figura 2.4 – Ilustração da descida agressiva [4].

2.2.3 Algoritmos Genéticos

Os algoritmos genéticos (GA) foram desenvolvidos por John Holland nos seus trabalhos em 1960 [6] e também por Goldberg [7]. Estes algoritmos foram inspirados na teoria de Darwin sobre a evolução natural, sendo um processo em que as sucessivas gerações de plantas e/ou animais vão guardando as melhores características dos antecessores, de forma a irem adoptando características cada vez mais adaptadas ao meio onde vivem [8].

Este é um método de genótipo, ou seja, as variáveis do problema são codificadas sob a forma de genes, correspondendo cada elemento a um cromossoma, uma sequência de bits que codificam o problema. Esta codificação implica a execução de um algoritmo descodificador para extrair o valor das variáveis naturais do problema. A Figura 2.5 apresenta o exemplo de uma codificação de um elemento cuja representação natural corresponde aos valores 2 e 5, que correspondem em binário a 0010 e a 0101, respectivamente.

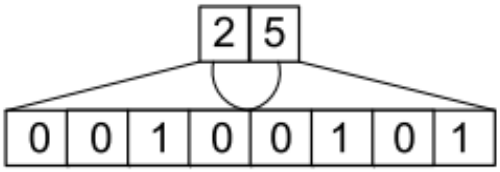

Figura 2.5 - Codificação de um gene [5].

Uma das questões a considerar em primeiro lugar é o tamanho da população, e segundo o método pelo qual os indivíduos são escolhidos. O tamanho da população tem sido abordado por vários pontos de vista teóricos, dando a ideia que existe um *trade-off* entre eficiência e eficácia. O tamanho da população não deve ser por isso muito pequeno, impedindo que houvesse espaço para explorar o espaço de pesquisa eficazmente, nem deve ser muito grande, o que impediria a eficiência do método num tempo razoável [2].

Cada geração guarda parte dos genes dos progenitores, mas sofre alterações de modo a que os elementos, cujas características se alteram no sentido de uma melhor adaptação ao meio, tenham maiores probabilidades de sobreviver e de procriar. Cada elemento é caracterizado por um conjunto de genes que são combinados entre os diferentes progenitores para gerar novos elementos [9].

Um gene é representado por um bit, sendo um cromossoma representado por um conjunto de bits. Para a procriação são permitidas operações de recombinação entre os progenitores, cruzando os bits num dado ponto de intersecção aleatório e mutação através da inversão de bits escolhidos aleatoriamente, situação que está ilustrada na Figura 2.6 [10].

Na fase de selecção são escolhidos os cromossomas que seguem para a próxima geração, recorrendo a uma regra de selecção que pode ser:

- Processo de selecção elitista, em que são seleccionados os melhores elementos entre a geração mutada e os progenitores;

- Torneio estocástico, em que, de entre dois elementos aleatórios, o mais forte é seleccionado com uma probabilidade p, e existe a probabilidade (1-p) de ser seleccionado o elemento mais fraco (tipicamente p=0.8). O objectivo deste factor aleatório é permitir uma pesquisa mais vasta no espaço de soluções;
- Selecção proporcional à aptidão (*fitness*). Nesta técnica existe uma probabilidade associada ao valor da aptidão de cada cromossoma sendo a escolha função de um valor probabilístico. Desta forma, é possível que algumas boas soluções sejam descartadas em detrimento de algumas piores, permitindo uma mais vasta pesquisa no espaço de soluções.

Podem-se observar na Figura 2.6 os processos inerentes à replicação e mutação dos cromossomas. Neste caso, dois cromossomas geram dois descendentes, através do cruzamento num ponto escolhido aleatoriamente. Depois, cada descendente sofre mutação em genes escolhidos de forma aleatória. Em seguida, são avaliados cada um dos novos cromossomas e seleccionados os melhores para a próxima geração.

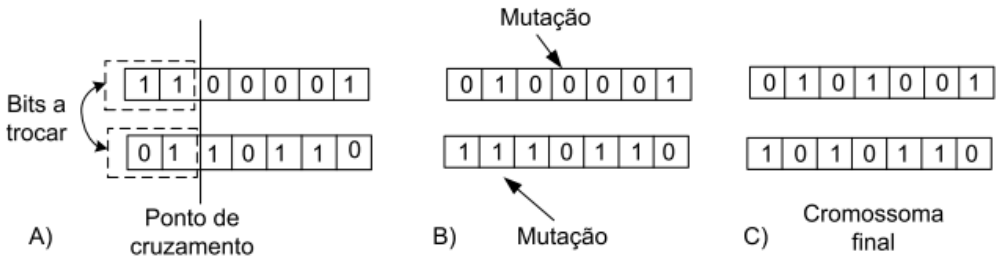

Figura 2.6 - Exemplo de sequência de alterações sofridas por dois cromossomas [5].

Deve haver especial atenção no que diz respeito ao espaço de pesquisa. A função de avaliação deve penalizar os cromossomas que se situem fora do espaço de pesquisa, baixando a probabilidade de esses serem seleccionados para a próxima geração. Outra opção para lidar com os limites de pesquisa, pode ser limitando a localização de novas partículas, isto é, após uma partícula ser criada e mutada, deve ser realizada uma nova mutação que introduza as partículas para o interior dos limites do espaço de soluções. Por exemplo, num problema de despacho económico, os geradores tem limites máximos e mínimos de produção [5].

Quando algum gerador ultrapassa o valor máximo permitido, deve-se penalizar a função objectivo. Por outro lado, na condição de que a produção tem que ser igual ao consumo mais as perdas, é preferível fazer um ajuste de forma proporcional, repartindo por todos os geradores a diferença entre a potência gerada e as cargas somadas das perdas [5].

Na Figura 2.7 é esquematizado um modelo de algoritmo genético apresentando todos os seus passos em forma de fluxograma.

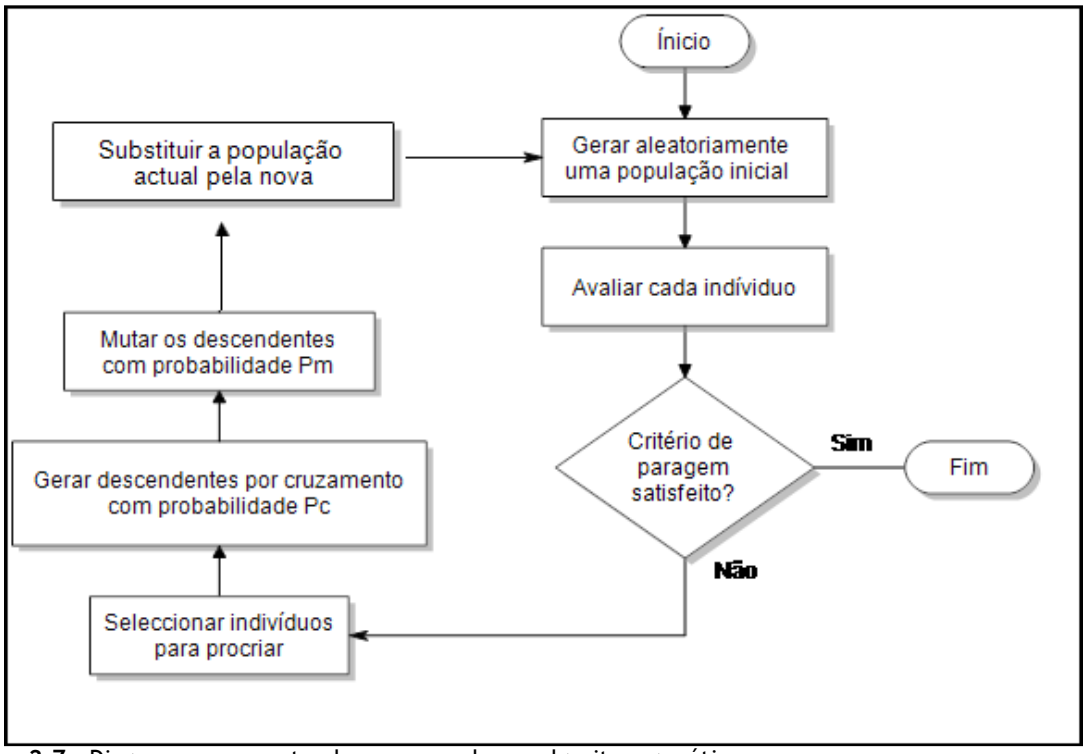

Figura 2.7 - Diagrama representando os passos de um algoritmo genético.

2.2.4 *Variable Neighborhood Search* (VNS)

Seja a expressão seguinte um problema de optimização:

$$\min \{f(x) | x \in X, X \subseteq S\} \tag{2.5}$$

S, X, x e f são o espaço de soluções, o conjunto de soluções admissíveis, uma solução admissível e uma função de valor real, respectivamente [2].

Variable Neighborhood Search (VNS), uma metaheurística proposta há poucos anos [11,12], é baseada num princípio simples: mudança sistemática de vizinhança durante a pesquisa. O seu desenvolvimento tem sido rápido, havendo muitos artigos já publicados que a utilizam. Têm sido feitas muitas extensões a esta técnica, principalmente para permitir a solução de problemas de grandes dimensões. Na maior parte deles tem sido feito um esforço para manter a simplicidade da esquema básico [2].

VNS é baseado em três simples considerações:

Consideração 1: Um mínimo local dizendo respeito a uma estrutura de vizinhança não está necessariamente acompanhado de outro mínimo local nessa mesma estrutura.

Consideração 2: Um mínimo global é um mínimo local no que diz respeito a todas as estruturas de vizinhança.

Consideração 3: Em muito problemas, os mínimos locais no que diz respeito a uma ou várias estruturas de vizinhança, N_k, são relativamente próximos entre eles.

Para resolver o problema indicado na expressão 2.5 usando várias vizinhanças, as considerações 1-3 podem ser usadas de três maneiras distintas: (i) determinística; (ii) estocática; (iii) ambas, determinística e estocástica. No que diz respeito à primeira estratégia, existe um método, *Variable Neighborhood Descent* (VND), que é utilizado quando a mudança entre vizinhanças for conseguida de um modo determinístico. Os passos deste método são apresentados de seguida [2].

Inicialização: Seleccionar o conjunto de vizinhanças de estrutura N_k, para k= 1, …., $k$máx, que pode ser usado na pesquisa; procurar uma solução inicial x; escolher uma condição de paragem;

Repetir a seguinte sequência até a condição de paragem ser verificada:

Colocar $k \leftarrow 1$;

Repetir os seguintes passos até k= $k$máx;

Sortear. Gerar um ponto x' aleatoriamente da $k$ésima vizinhança de x ($x' \in N_k(x)$);

Mover ou não. Se este ponto é melhor que o histórico, mover para ($x \leftarrow x'$), e continuar a pesquisar com $N_1(k \leftarrow 1)$; senão, colocar $k \leftarrow k+1$;

A maioria das heurísticas de pesquisa usa uma ou às vezes duas vizinhanças ($k'_{max} \leq 2$). De notar que a solução final deve ser um mínimo local no que diz respeito a todas as vizinhanças, e, portanto, as hipóteses de alcançar o mínimo global são maiores do que se fosse utilizada apenas uma estrutura [2].

A segunda forma (estocática) contém um modelo ao qual é chamado de VNS reduzido (RVNS) e é obtido se os pontos aleatórios forem seleccionados de $N_k(x)$, sem seguirem a sua ascendência. Os seus passos são semelhantes aos do primeiro método [2].

A técnica RVNS é prática para problemas de grande dimensão. É observado que o melhor valor para o parâmetro $k_{máx}$ é normalmente 2. A juntar a isso, o máximo número de iterações entre duas melhorias é normalmente usado como critério de paragem. RVNS é semelhante ao método de *Monte-Carlo*, mas mais sistemático [2].

O método que combina a forma estocástica e determinística de mudar a vizinhança [12] é usualmente chamado de VNS básico. Os seus passos são indicados de seguida.

Inicialização: Seleccionar o conjunto de vizinhanças de estrutura N_k, para k= 1, …., $k$máx, que pode ser usado na pesquisa; procura uma solução inicial x; escolher uma condição de paragem;

Repetir a seguinte sequência até a condição de paragem ser verificada:

(1)Colocar $k \leftarrow 1$;

(2)Repetir os seguintes passos até k= $k$máx;

(a) Sortear. Gerar um ponto x' aleatoriamente da $k$ésima vizinhança de x ($x' \in N_k(x)$);

(b) Pesquisa Local. Aplicar um método de pesquisa local com x' como solução inicial; designar de x'' o óptimo local obtido;

(c) Mover ou não. Se este ponto é melhor que o histórico, mover para ($x \leftarrow x''$), e continuar a pesquisar com $N_1(k \leftarrow 1)$; senão, colocar $k \leftarrow k+1$;

O critério de paragem deve ser, por exemplo, o máximo de tempo permitido pelo CPU, o máximo número de iterações, ou máximo número de iterações entre duas melhorias. Frequentemente vizinhanças sucessivas encaixam-se uma na outra. De notar que há um ponto gerado aleatoriamente no passo 2 de forma a evitar movimentos cíclicos, que podem ocorrer se alguma regra determinística for usada [2].

2.2.5 *Guided Local Search* (GLS)

Local Search (LS) é a base da maior parte das técnicas heurísticas de pesquisa. Consegue encontrar soluções muito rapidamente. No entanto, pode cair na armadilha que são os óptimos locais. Para melhorar a eficácia do LS, várias técnicas têm sido criadas ao longo dos anos como *Simulated Annealing* (SA), *Tabu Search* (TS) e *Guided Local Search* (GLS). Nesta secção será abordada esta última.

GLS é um algoritmo metaheurístico generalizado e alargado de uma rede neuronal baseada num método chamado de GENET. GENET é um método ponderado para satisfação de restrições. Para aplicar o método GLS, primeiro define-se um conjunto de características para cada solução candidata. Quando o LS é apanhado em óptimos locais, certos atributos são seleccionados e penalizados. As pesquisas LS usam a função objectivo para acumular todas as penalidades. A novidade no GLS está relacionada com a forma como são seleccionadas as características a penalizar. GLS distribui eficazmente o esforço da pesquisa pelo espaço de pesquisa, favorecendo áreas mais promissoras em relação à melhoria das características [2].

Para aplicar esta técnica é necessário definir as características do problema. É associado um custo e uma penalidade a cada característica. Os custos podem ser definidos pelos termos da função objectivo e seus coeficientes. As penalidades são colocadas a 0 e só são

incrementadas se a pesquisa alcançar um óptimo local. Dada uma função objectivo *g* que mapeia todas as soluções candidatas *s* para um valor numérico, GLS define uma função h que será usada pela pesquisa local (substituindo g):

$$h(s) = g(s) + \lambda \times \sum (p_i \times I_i(s)) \tag{2.6}$$

Nesta expressão *s* é a solução candidata, λ é um parâmetro do algoritmo GLS, *i* é o número do atributo, p_i é a penalidade para a característica *i* e I_i é uma indicação se *s* apresenta a característica *i*, valendo 1 se apresentar e 0 caso contrário. A intenção consiste em penalizar as características menos favoráveis ou características que "mais importam" quando a pesquisa local encontra um óptimo local. A característica que tem um custo associado mais elevado afecta mais o custo total. Outro factor a ter em conta é o valor da penalidade corrente dessa característica. A utilidade de penalizar a característica *i*, $util_i$, com um óptimo local s_*, é definida pela expressão (2.7):

$$util_i(s_*) = I_i \times \frac{c_i}{1 + p_i} \tag{2.7}$$

Nesta expressão c_i é o custo e p_i é o valor da penalidade corrente da característica *i*. Por outras palavras, se o óptimo local não apresentar uma determinada característica, significa que a utilidade de penalizar essa característica é nula. Quanto maior o custo da característica (maior c_i), maior será a utilidade de a penalizar. Porém, quanto mais vezes essa característica for penalizada (maior p_i), menor será a utilidade de a penalizar de novo. Num óptimo local, a característica com o maior valor de utilidade será penalizada. Quando uma característica é penalizada, o seu valor de penalidade é sempre incrementado de uma unidade. A escala de penalidade é ajustada por λ [2].

Tendo em conta o custo e a penalidade corrente em consideração na selecção da característica a penalizar, GLS foca o seu esforço de pesquisa nas áreas mais promissoras do espaço de pesquisa, ou seja áreas que contêm soluções candidatas que exibem características que traduzem baixo custo. Por outro lado, as penalidades servem para prevenir a pesquisa de direccionar todo o seu esforço para uma região particular do espaço de soluções [2].

Em seguida é apresentado o procedimento geral do GLS. *L* é uma estratégia LS, *g* é uma função objectivo, *I* e *c* são vectores com custos e atributos e λ é um valor numérico.

Procedimento GLS (L, g, λ , I, c)

1. Gerar soluções candidatas iniciais aleatoriamente;
2. Inicializar todos os valores das penalidades, pi=0;

3. Repetir os passos seguintes até critério de paragem ser satisfeito (por exemplo número de iterações sem que a função objectivo tenha melhorado mais do que uma percentagem a definir):

3.1 Realizar uma pesquisa local (usando L) de acordo com a função h até um óptimo local ser atingido;

3.2 Para cada característica *i* exibida calcular a utilidade $util_i$;

3.3 Penalizar todas as características *i* cuja $util_i$ seja máxima. *pi=pi+1*

4. Retornar a melhor solução candidata encontrada até ao momento de acordo com a função objectivo g.

Fim do procedimento GLS

Orientações para implementar um pseudo-código GLS

Método de construção

Tal como outras metaheurísticas, o GLS requer um método de construção para gerar uma solução inicial para o problema. No pseudo-código, é denotado como *ConstructionMethod*. Este método pode gerar uma solução aleatória ou uma solução heurística baseada em alguma técnica para construir soluções para um problema particular [2].

Método de melhoramento

É igualmente necessário dispor de um método para melhorar a solução. No pseudo-código, este método é denotado por *ImprovementMethod*. Este método pode ser um simples algoritmo de pesquisa local ou um mais sofisticado tal como VNS ou combinações de métodos de pesquisa local com algoritmos de pesquisa clássicos [2].

Não é essencial que o método de melhoramento gere mínimos locais de alta qualidade. Experiências com GLS e outras heurísticas revelam que esses mínimos demoram um certo tempo a ser alcançados. De notar ainda que o método de melhoramento é usado com a função objectivo aumentada (*h(s)*) em vez da original (*g(s)*) [2].

Funções indicativas e penalização de características

Estando disponíveis um método de construção e um método de melhoramento, o pseudo-código está pronto a aplicar. As penalidades das características são inicializadas a zero e são incrementadas para características que maximizam a utilidade, depois de cada chamada do método de melhoramento. As funções indicativas I_i para as características raramente precisam de ser implementadas. Tendo em conta os valores das variáveis de decisão é

possível identificar directamente as características presentes num mínimo local. A selecção de características a penalizar pode ser eficientemente implementada usando o mesmo ciclo para calcular a utilidade para as características presentes no mínimo local e também para colocar características com utilidade máxima num vector. Com o segundo ciclo, as características com utilidade máxima contidas no vector têm as suas penalidades incrementadas numa unidade [2].

O parâmetro λ e critério de paragem

O valor do parâmetro λ do modelo GLS é difícil de encontrar. Felizmente, para muitos problemas, tém-se verificado que bons valores para λ podem ser encontrados dividindo o valor da função objectivo num mínimo local pela média de características presentes.

Há muitas escolhas possíveis para o critério de paragem. Desde que o GLS não seja apanhado em mínimos locais, não existe um ponto óbvio em que se deva parar o algoritmo. Algumas possibilidades para este critério residem portanto em escolher um número limite de movimentos, número limite de movimentos avaliados, ou mesmo tempo máximo dispendido pelo CPU a executar o algoritmo. Estes critérios não garantem que está encontrada uma boa solução, embora já tenham sido realizadas um número de iterações razoável ou um tempo de execução longo. Por isso, é preferível utilizar como critério de paragem um número de iterações máximo sem que a função objectivo melhore mais do que uma percentagem a definir. [2]

2.2.6 *Greedy Randomized Adaptive Search Procedures* (GRASP)

O GRASP (*Greedy Randomized Adaptive Seach Procedure*) [13,14] é um método *multi-start* aperfeiçoado que incorpora um processo iterativo, em que cada iteração inclui duas fases: construção e pesquisa local. A fase de construção constrói soluções admissíveis, sendo a sua vizinhança investigada até um mínimo local ser encontrado durante a fase pesquisa local. A melhor solução global é mantida como resultado. O próximo procedimento mostra o pseudo-código referente ao procedimento GRASP, no qual *Max_iterações* representa o número de iterações são realizadas e *Semente* é usada como primeira semente para geração de números pseudo aleatórios [2].

Procedimento GRASP(Max_iterações, Semente)
1 Ler Entrada();
2 **Para** k=1, …, Max_iterações **fazer**

3 Solução ← Greedy_Randomized_Construção(Semente);
4 Solução ← Pesquisa_Local(Solução);
5 Actualiza_solução(Solução,Melhor_solução);
6 **Fim**;
7 **Retorna** Melhor_solução
Fim GRASP

O próximo procedimento ilustrado corresponde ao pseudo-código da fase de construção. As soluções candidatas são formadas por todos os elementos que podem ser incorporados na solução parcial sob construção, mas de forma a não destruir a sua viabilidade. A selecção do elemento seguinte para incorporação é determinada pela avaliação de todos os elementos candidatos de acordo com função de avaliação *greedy*. Esta função usualmente representa o incremento da função custo dada a incorporação deste elemento na solução em construção. A avaliação dos elementos por esta função leva à criação de uma Lista de Candidatos Restrita (RCL) formada pelos melhores elementos, i.e. aqueles cuja incorporação na solução parcial corrente resulta em pequenos custos incrementais (este é o aspecto "ganancioso" - *greedy* - do algoritmo). O elemento a ser incorporado na solução parcial é aleatoriamente seleccionado de entre os elementos da RLC (esta é parte probabilística da heurística). Uma vez seleccionado, o elemento é incorporado na solução parcial, a lista de candidatos é actualizada e os custos incrementais são reavaliados (esta é a parte adaptativa da heurística) [2].

Procedimento Greedy_Randomized_Construção(Semente)
1 *Solução* ← 0;
2 Avaliar os custos incrementais dos elementos candidatos;
3 **Enquanto** *Solução* não for uma solução completa **fazer**
4 Construir a lista restrita de candidatos (RCL);
5 Seleccionar um elemento *s* da RCL aleatoriamente;
6 *Solução* ← *Solução* $\cup \{s\}$;
7 Reavaliar os custos incrementais;
8 **Fim**;
9 **Retorna** Solução;
Fim Greedy_randomized_Construção

As soluções geradas pela fase de construção não são necessariamente óptimas, mesmo com vizinhanças simples. A fase de pesquisa local normalmente melhora a solução construída. O algoritmo de pesquisa local substitui sucessivamente a solução corrente por uma solução melhor na vizinhança da solução corrente. Este procedimento termina quando não for

encontrada nenhuma solução melhor na vizinhança. O pseudo-código do algoritmo da pesquisa local começa com a solução *Solução* construída na fase anterior e usando uma vizinhança *N* sendo apresentado a seguir [2].

Procedimento Pesquisa_Local(*Solução*)
1 **Enquanto** *Solução* não for óptima localmente **fazer**
2 *Encontrar s' ∈ N(Solução) com f(s') < f(Solução);*
3 *Solução ← s'*;
4 **Fim**;
5 **Retornar** *Solução*;
Fim Pesquisa_Local

Apresentam-se em seguida alguns comentários importantes sobre esta técnica em jeito de conclusão:

- É de fácil implementação, dada uma heurística ambiciosa e um procedimento de pesquisa local [4];
- Alguns parâmetros como número de iterações e lista de candidatos podem ser fixos [4];
- O GRASP precisa de boas soluções iniciais para ter sucesso nos seus resultados, ao contrário de, por exemplo, a técnica *Tabu Search*[4];
- Podem ser usados filtros para a aceleração da pesquisa local [4].

2.2.7 Colónia de Formigas – *Ant Colony Optimization* (ACO)

A optimização utilizando Colónia de Formigas (ACO) [15,16] é uma abordagem recente para resolver problemas de optimização altamente combinatórios. A fonte de inspiração deste método é o rasto de feromonas, baseado no comportamento de formigas reais que usam feromonas como meio de comunicação entre elas, o que parece ajudá-las a encontrar facilmente o caminho entre a colónia e uma fonte de comida [4]. Analogamente, ACO é baseado na comunicação indirecta de uma colónia de simples agentes, chamados de formigas (artificiais), mediada por rastos de feromonas (artificiais). Na ACO estes rastos servem como uma informação numérica distribuída que as formigas usam para construir probabilisticamente soluções, de modo a resolverem o problema e a permitir que as formigas se adaptem durante a execução do algoritmo para reflectir a experiência obtida durante o processo de pesquisa. A Figura 2.8 apresenta uma ilustração deste método [2].

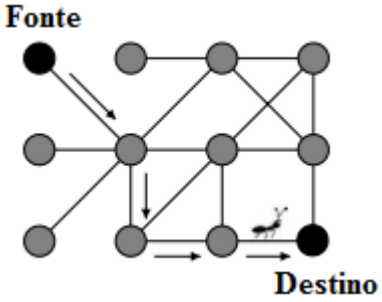

Figura 2.8 – Ilustração do algoritmo utilizando Colónias de Formigas [17].

A cooperação é a palavra-chave neste método, onde os recursos computacionais são alocados por agentes relativamente simples (formigas artificiais) que comunicam indirectamente. As boas soluções são uma propriedade emergente da interacção dos agentes comunicativos. Os algoritmos ACO podem ser usados tanto em problemas de optimização combinatórios estáticos como dinâmicos [17].

Uma formiga artificial no ACO corresponde a um procedimento construtivo estocástico que, de forma incremental, constrói uma solução, adicionando componentes a uma solução parcial sob construção. Depois, o ACO pode ser aplicada a qualquer problema de optimização combinatório, para o qual uma heurística construtiva possa ser definida [17].

Vamos considerar agora um problema de minimização (S, f, Ω), onde S é o conjunto de soluções candidatas, f é a função objectivo que atribui um valor numérico $f(s,t)$ a cada solução candidata $s \in S$ e $\Omega(t)$ é o conjunto de restrições. O parâmetro t indica que a função objectivo e as restrições podem ser dependentes do tempo, por exemplo, em aplicações de problemas dinâmicos. O objectivo a atingir consiste em encontrar a solução globalmente óptima s^* que corresponda à solução de custo mínimo para um problema de minimização. O problema de optimização combinatória (S, f, Ω) que é mapeado no problema é caracterizado por este conjunto de itens [17]:

- Um conjunto finito $C=\{c_1,c_2, \ldots, C_{Nc}\}$ de componentes é dado, onde Nc é o número de componentes;
- Os *estados* do problema são definidos em termos de sequências $x= <c_i, c_j, \ldots c_h,..>$ de comprimento finito sobre elementos de C. O conjunto de todas as possibilidades de estados é denotada por χ. O comprimento da sequência x, que é o número de componentes na sequência, é expresso por |x|. O máximo comprimento da sequência é dada por uma constante n;
- O conjunto das soluções (candidatas) S é um subconjunto de χ, i.e., $S \subseteq \chi$.
- Um conjunto de estados viáveis χ' com $\chi' \subseteq \chi$, definidos através de um teste dependente do problema, que verifica que é possível completar a sequência $x \in \chi'$ numa solução que satisfaça as restrições Ω;
- Um conjunto não vazio S^* de soluções óptimas, com $S^* \subseteq \chi'$ e $S^* \subseteq \chi$;

- Um custo $g(s,t)$ é associado a cada solução candiadata $s \in S$. Na maior parte dos casos $g(s,t) \equiv f(s,t), \forall s \in S'$, onde $S' \subseteq S$ é o conjunto de soluções candidatas viáveis obtidas de S através das restrições Ω(t);
- Em alguns casos o custo, ou a estimativa do custo, $J(x,t)$, pode estar associado a estados, excepto soluções candidatas. Se x_j pode ser obtido adcionando componentes de soluções a um estado x_i , então $J(x_i, t) \leq J(x_j,t)$. Notar que $J(s,t) \equiv g(s,t)$.

Dada esta formulação, as formigas artificiais constroem soluções realizando percursos aleatórios num grafo completamente ligado $G_C = (C, L)$ onde os nós correspondem aos componentes C e o conjunto L liga completamente os componentes C. O grafo G_C é chamado *grafo de construção* e os elementos de L são chamados de *conexões* [17].

As restrições do problema Ω(t) são implementadas na heurística construtiva das formigas. Na maior parte das aplicações, as formigas constroem soluções viáveis. No entanto, às vezes é necessário ou benéfico que elas construam soluções não viáveis. Os componentes $c_i \in C$ e conexões $l_{ij} \in L$ podem ter associado um rasto de feromonas τ (τ_i se associado com componentes, τ_{ij} se associado com conexões), e um valor heurístico η (η_i e η_{ij}, respectivamente). O rasto das feromonas codifica uma memória de longo prazo acerca do processo de pesquisa da formiga, e é actualizado pelas próprias formigas. Aliás, o valor heurístico representa uma informação prioritária acerca do problema ou informações de tempo de execução fornecidas por uma fonte diferente das formigas. Em muito casos, η é o custo, ou uma estimativa do custo, de adicionar o componente ou conexão à solução sob construção. Estes valores são usados pela regra da heurística das formigas para tomar decisões probabilísticas sobre como se mover no grafo. Mais precisamente, cada formiga k da colónia tem as seguintes propriedades [17]:

- Explora o grafo de construção;
- Tem uma memória M^k onde podem guardar informação sobre o caminho que seguiram até ao momento. A memória pode ser usada para criar soluções viáveis, calcular valores heurísticos, avaliar a solução encontrada e remarcar o caminho de volta;
- Tem um estado inicial x_s^k e um ou mais critérios de paragem e^k;
- Quando no estado $x_r = \langle x_{r-1}, i \rangle$, se nenhum critério de paragem for satisfeito, move-se para o nó j na sua vizinhança, que é, para o estado $\langle x_r, j \rangle \in \chi$. Se pelo menos um dos critérios for satisfeito, então a formiga pára;
- Selecciona um movimento aplicando uma regra da decisão probabilística. Essa regra é uma função dos rastos de feromonas e valores heurísticos, da memória privada das formigas do estado corrente e das restrições do problema;

- Quando adiciona um componente c_j ao estado corrente, pode adaptar o rasto das feromonas associadas com a conexão correspondente;
- Uma vez construída a solução, pode ser remarcado o caminho de regresso e ser actualizado o rasto das feromonas de componentes do grafo que foram usados.

É importante notar que as formigas agem independentemente e, embora cada formiga seja suficientemente complexa para encontrar uma solução (provavelmente pobre) para o problema em consideração, soluções de boa qualidade só podem emergir como resultado de interacção colectiva entre as formigas. Isto é obtido através da comunicação indirecta mediada por informação lida ou escrita pelas formigas nas variáveis de armazenamento de valores dos rastos de feromonas. De certo modo, este é um processo de aprendizagem distribuído no qual agentes simples, formigas, não são elas próprias adaptativas mas, pelo contrário, adaptativamente modificam a forma como o problema é representado e são percebidas pelas outras formigas [17].

Pode-se dizer que um algoritmo ACO resulta da acção combinada de três procedimentos: *Formigas_Constroem_Soluções, Actualiza_Feromonas* e *Acções_Centrais*.

O primeiro procedimento controla a colónia de formigas que concorrentemente visitam estados adjacentes do problema considerado movendo-se através da vizinhança do grafo de construção G_C. O segundo procedimento representa o processo pelo qual os rastos das feromonas são modificados. Finalmente *Acções_Centrais* é usado para implementar acções centralizadas que não podem ser efectuadas por simples formigas. Exemplos dessas acções são a activação do procedimento de optimização local, ou a obtenção de informação global que possa ser usada para decidir se é útil ou não depositar feromonas adicionais para influenciar o processo de pesquisa de uma perspectiva não local [17].

De seguida é apresentado o pseudo-código referente a um algoritmo ACO, usando estes três procedimentos que foram referidos anteriormente.

```
Procedimento Metaheuristica_ACO
    Lista_Actividades
        Formigas_Controem_Soluções
        Actualiza_Feromonas
        Acções_Centrais          % opcional
    Fim_Lista_Actividades
Fim_Procedimentos
```

2.2.8 *Simulated Annealing (SA)*

2.2.8.1 Analogia e considerações iniciais

Simulated Annealing (SA) é uma das técnicas mais flexíveis para resolver problemas combinatórios complexos. A maior vantagem do SA é a de poder ser aplicado a problemas de grandes dimensões independentemente das condições de diferenciação, continuidade e convexidade que são normalmente necessárias nos métodos convencionais de optimização [18].

Annealing (recozimento) é o processo de submeter um sólido a uma temperatura elevada, com consequente arrefecimento, de forma a obter cristais de alta qualidade (i.e. cristais cuja estrutura forme reticulados perfeitos). O *Simulated Annealing* simula o processo físico do *recozimento* e foi originalmente proposto no domínio de mecanismos estáticos de modo a modelar o processo natural de solidificação e formação dos cristais. Durante o processo de arrefecimento, é assumido que as condições de equilíbrio térmico se mantêm. O processo de arrefecimento termina quando o material atinge um estado de energia mínima que, em princípio, corresponde um cristal perfeito. É sabido que os cristais sem defeitos (i.e com energia mínima) são mais susceptíveis de serem formados sob um processo de arrefecimento lento. As duas maiores características do processo do *Simulated Annealing* são o mecanismo de transição entre estados e o arrefecimento. Quando aplicado à optimização de problemas combinatórios, este método pretende encontrar uma configuração óptima (ou estado com energia mínima) de um problema complexo [18].

Simulated Annealing foi originalmente proposto por Metropolis no princípio da década de 1950 como um modelo do processo de cristalização. Foi só em 1980, contudo, que uma pesquisa independente realizada por Kirkpatrick, Gelatt, e Vecchi [19] e Cerny notaram as similaridades entre o processo físico do recozimento e alguns problemas de optimização combinatórios. Notaram que havia uma correspondência entre estados físicos da matéria alternativos e o espaço de soluções de um problema de optimização. Observaram também que a função objectivo de um problema de optimização poderia corresponder à energia libertada do material. Uma solução óptima é associada a um cristal perfeito, ao passo que um cristal com defeitos corresponde a uma solução óptima local [18]. A Figura 2.9 mostra esta analogia e alguns defeitos comuns nos cristais.

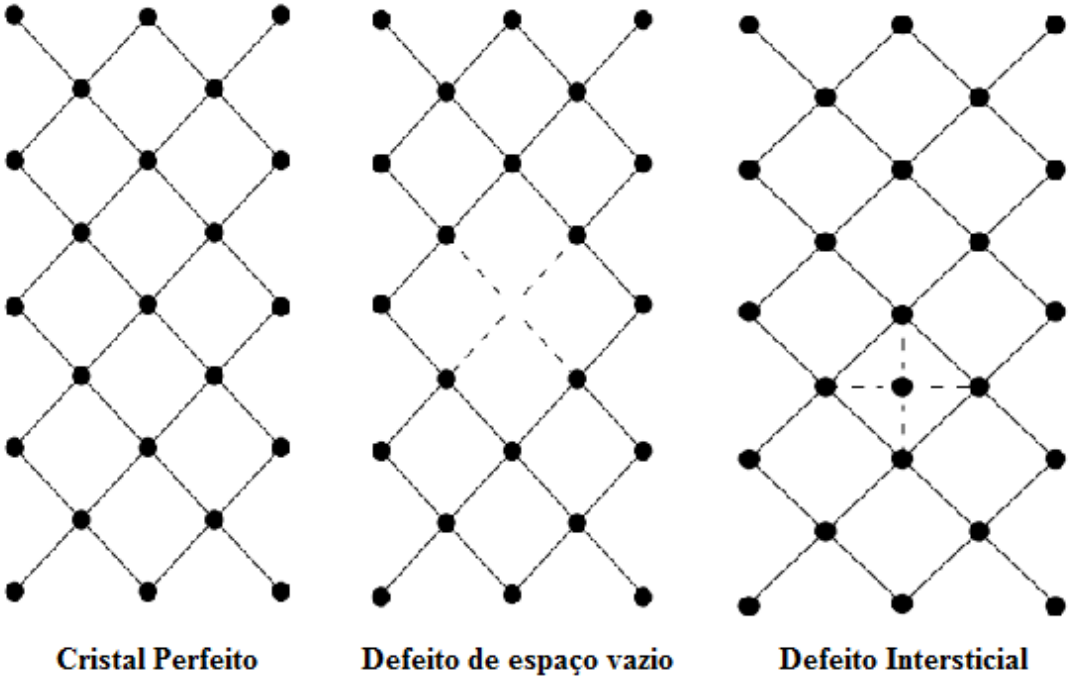

Figura 2.9 – Processo de cristalização e defeitos mais usuais.

A analogia não está ainda completa, no entanto, porque no processo de recozimento existe uma variável física que é a temperatura, associada à qual é direccionado um controlo apropriado de forma obter o cristal perfeito. Quando o *Simulated Annealing* é usado como técnica de optimização, a "temperatura" torna-se num simples parâmetro de controlo que tem de ser determinado de forma própria de forma a alcançar os resultados desejados [18].

2.2.8.2 Algoritmo de Metropolis

A ideia original associada ao algoritmo *Simulated Annealing* é o algoritmo Metropolis que modeliza o comportamento microscópico de conjuntos enormes de partículas, como num sólido, usando a simulação de *Monte Carlo*. Num material, as partículas têm diferentes níveis de energia, de acordo com uma certa distribuição satisfatória. O nível mais baixo de energia possível, conhecido como nível fundamental, corresponde ao estado em que todas as partículas ficam estáveis e ocorre à temperatura 0 K. Para temperaturas acima desse nível, as partículas irão ocupar diferentes níveis de energia, de tal forma que o número de partículas em cada nível decresce tal como o nível de energia decresce (i.e o número máximo de partículas é encontrado no nível fundamental). A distribuição das partículas nos vários níveis varia com a temperatura. Para T= 0 K, por exemplo, todas as partículas estão no estado fundamental. À medida que a temperatura aumenta, mais partículas são encontradas em níveis de energia mais elevados mas sempre com uma função decrescente de nível de energia [18].

O algoritmo Metropolis gera uma sequência de estados de um sólido como o descrito: dado um sólido num estado S_i, com energia E_i, o próximo estado S_j é gerado por um mecanismo de transição que consiste numa pequena perturbação no que diz respeito ao estado original, obtido movendo uma das partículas de um sólido escolhida por um processo de sorteio. Seja a energia do estado resultante, também encontrado aleatoriamente, E_j; se a diferença E_j - E_i for menor ou igual a zero, o novo estado S_j é aceite. Senão, no caso da diferença ser maior que zero, o novo estado é aceite com a probabilidade dada por 2.8.

$$P_{aceitação} = \exp^{\left(\frac{E_i - E_j}{K_B T}\right)} \tag{2.8}$$

Nesta expressão T é a temperatura do sólido e K_B é a constante de *Boltzman*. Esta regra de aceitação é também conhecida como critério Metropolis e o algoritmo sumarizado em cima é o algoritmo de Metropolis. A temperatura é assumida como tendo uma taxa de variação tal que o equilíbrio termodinâmico é alcançado para um nível de temperatura corrente, antes de se mover para o nível seguinte. Isto normalmente requer um número grande de transições de estado do algoritmo de Metropolis [18].

2.2.8.3 Algoritmo *Simulated Annealing*

Para um problema de optimização combinatório ser resolvido pelo *Simulated Annealing*, consideremos os elementos seguintes: seja *G* um conjunto finito de configurações e *v* o custo associado de cada configuração de *G*. A solução deste problema combinatório consiste em pesquisar o espaço de configurações para o par (*G*, *v*) que apresente menor custo. O algoritmo SA começa com uma configuração inicial G_0 e uma "temperatura" inicial $T = T_0$ e gera uma sequência de configurações $N = N_0$. Depois a Temperatura é diminuída, é determinado o novo número de iterações a ser realizado no nível de temperatura, e o processo é então repetido. Uma configuração candidata é aceite se o seu custo é menor que o da configuração corrente. Se o custo da configuração candidata é maior que o custo da configuração corrente, pode ainda ser aceite com uma determinada probabilidade. Isto deve-se à pretensão de escapar de configurações óptimas locais, saltando para outras soluções piores. Todo o processo é controlado por uma sequência *de arrefecimento* que determina como a temperatura é diminuída durante o processo de pesquisa [18].

De seguida será sumariamente apresentado o algoritmo SA, que consiste em dois mecanismos básicos: a geração de alternativas e a regra de aceitação.

Procedimento Simulated_Annealing;

Início

Inicialização (T_0, N_0);

K:=0;

Configuração inicial S_i

Repetir Procedimento

Fazer *L*:=1 até N_k

Gerar (S_j a partir de S_i);

Se $f(S_j) \leq f(S_i)$ fazer $S_i := S_j$;

Senão

Se $\exp\left(\frac{f(S_i)-f(S_j)}{T_k}\right)$ > Aleatório [0,1] **fazer** $S_i := S_j$;

Fim do fazer;

K: =k+1;

Cálculo do comprimento (N_k)

Determinar parâmetro de controlo (T_k)

Critério de paragem;

Fim

T_k é o parâmetro de controlo que corresponde analogamente à temperatura no processo de arrefecimento físico, e N_k é o número de alternativas geradas no *k-ésimo* nível de temperatura. Inicialmente, quando T é grande, é permitida uma maior deterioração da função custo. À medida que a temperatura decresce, o algoritmo SA torna-se mais "ganancioso", e só pequenas deteriorações são aceites. Finalmente, quando *T* tende para zero, já não há deteriorações aceites [18].

Para o estado corrente S_i, com custo $f(S_i)$, uma solução vizinha S_j, com custo $f(S_j)$, é gerada pelo mecanismo de transição sendo calculada a probabilidade dada por (2.9) em conformidade com o teste de aceitação.

$$P_T\{AceitarS_j\} = \begin{cases} 1 & se \quad f(S_j) \leq f(S_i) \\ \exp\left(\frac{f(S_i)-f(s_j)}{T_k}\right) & se \quad f(S_j) > f(S_i) \end{cases} \tag{2.9}$$

Há que ter em conta vários parâmetros ligados ao controlo do arrefecimento, ou seja à estratégia usada desde o início até à convergência do algoritmo SA. Esses aspectos são:

- Temperatura inicial T_0;
- Temperatura final T_f (critério de paragem);
- Número de transições, N_k, à temperatura T_k;

- Esquema de redução de temperatura

A eficiência do algoritmo, em relação à qualidade das soluções finais assim como ao número de iterações, dependerá das escolhas destes parâmetros [18].

2.2.8.4 Critério de Paragem

O critério de paragem varia muito com a complexidade e sofisticação do problema. Algumas das estratégias mais comuns são definidas em seguida. Uma delas consiste em definir um número constante de reduções de temperatura normalmente entre 6 e 50. Outra abordagem consiste em usar uma taxa de melhoria da função objectivo para definir o critério de paragem. Uma vez aí, se o custo da melhor solução até ao momento não melhorar pelo menos uma margem especificada depois de uma série de reduções de temperatura, é assumido que a convergência foi alcançada e o processo é parado. Há ainda outra abordagem que define um número de transições para piores soluções que podem ser aceites. O processo pára quando o número de aceitações se torne menor que um valor especificado.

2.2.9 Comentários

Os métodos usualmente denominados metaheurísticas são muito interessantes, pois permitem encontrar boas soluções num tempo muito razoável de pesquisa não garantindo, no entanto, que seja alcançado o óptimo global. Em problemas combinatórios muito complexos, problemas da vida real, a utilização dos métodos clássicos poderia revelar-se impossível, visto que o processamento de pesquisa poderia ser demasiado pesado. Assim, as metaheurísticas são as técnicas indicadas para resolver problemas de optimização de natureza combinatória, sendo que para isso há que saber formular e interpretar o problema de uma forma convenientemente ajustada à técnica a utilizar.

Capítulo 3

Análise bibliográfica de alguns modelos utilizando *Simulated Annealing*

Neste capítulo, serão descritos alguns trabalhos e projectos que utilizam o algoritmo *Simulated Annealing*. Houve a preocupação de seleccionar trabalhos que estão de alguma forma relacionados com Engenharia Electrotécnica, em especial a área de Energia. No entanto este algoritmo é extensível a muitas outras áreas. Assim, não se pretendeu enumerar e descrever de forma exaustiva todos os problemas descritos na literatura e utilizando *Simulated Annealing,* mas apenas seleccionar alguns deles como forma de ilustrar esta técnica. Os exemplos seguintes abordam o problema do *Unit Commitment*.

3.1 Simulated Annealing para o problema do Unit Commitment

O problema do *Unit Commitment* possui carácter combinatório e tem como objectivo o planeamento da operação dos geradores da rede. O problema de decisão baseia-se em distinguir os geradores que estarão em serviço (*on*) e os que não estarão (*off*) e durante quanto tempo estarão nesses estados (*on*/*off*). Consiste num planeamento normalmente entre dois dias a duas semanas, divididos em períodos horários. A escolha das unidades produtoras deve satisfazer a carga prevista do sistema e exigências de reserva, ao mínimo custo de operação, sujeito a restrições técnicas e ambientais. Devido a esta natureza combinatória, obter uma solução óptima é por vezes computacionalmente difícil ou impossível. Por isso, as heurísticas têm sido muito utilizadas de forma a encontrar uma solução satisfatória (não necessariamente óptima) [20].

3.1.1 Primeiro exemplo

A. Aspectos Gerais

Um exemplo interessante é o descrito em [20]. É usual na resolução do problema do *Unit Commitment* usando metaheurísticas representar a solução através de uma matriz binária, onde cada coluna representa a calendarização de cada unidade no horizonte de planeamento. Cada elemento dessa coluna representa o estado de uma unidade num determinado período. Esta é uma abordagem interessante, no entanto há a possibilidade de levar a soluções incompatíveis.

Em [20] é proposta outra abordagem que é representada na Figura 3.1. Começa-se por representar através de um vector a solução na sua forma tradicional para cada unidade de produção, onde 1's significam que a unidade está activa e 0's significam que a unidade está inactiva.

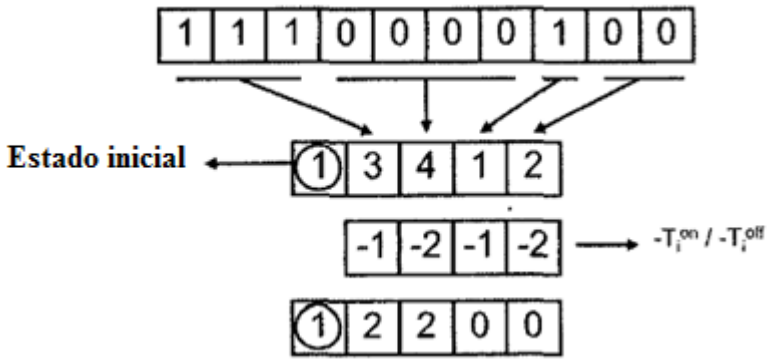

Figura 3.1 - Abordagem para o problema *Unit Commitment* [20].

Posteriormente é criado um novo vector que agrupa o número de vezes consecutivas que uma unidade de produção está no mesmo estado, sendo que o primeiro elemento do vector representa a solução inicial. Para finalizar, a cada elemento de cada coluna, excepto o primeiro, é subtraído o factor (T_i^{on}/T_i^{off}), onde:

- T_i^{on} é o tempo mínimo que a unidade de produção *i* deve estar ligada;
- T_i^{off} é o tempo mínimo que a unidade de produção *i* deve estar desligada.

A estrutura que daí resulta é particularmente adequada por permitir verificar as restrições de tempo mínimo no estado *ligado* ou no estado *desligado*. Se cada elemento do vector final tiver valor igual ou superior a zero, as restrições são cumpridas.

B. Estruturas de vizinhança

Foram criadas duas estruturas de vizinhança, *neighb1* e *neighb2*. A primeira começa por sortear aleatoriamente uma unidade de produção *i* e um período *t* (*t* é o ponto de transição de estado da unidade *i*) e:

1. Muda o estado da unidade *i* no período *t*; ou
2. Muda o estado da unidade *i* para todos os períodos entre *t* e a transição seguinte *t1*, se o intervalo entre *t* e *t1* for menor que um determinado valor, *n*.

A segunda estrutura tem como propósito reprimir estados de curta duração para uma determinada unidade. Para aplicar esta estrutura no algoritmo de solução há que ajustar um parâmetro S_d e também implementar um mecanismo que seja capaz de determinar situações em que esta alternativa não deva ser utilizada.

C. Estratégias de pesquisa

Para tornar o algoritmo mais eficientes foram definidas estratégias de pesquisa. A primeira estratégia começa por dividir o horizonte de planeamento em intervalos de acordo com a carga prevista para cada período de tempo. Depois, são aplicadas algumas regras à solução actual. Por exemplo, se uma unidade *i* está ligada no período *t*, num intervalo de carga normal, e a diferença entre *t* e as horas de pico é menor que *n* períodos, então se *i* está ligada nas horas de pico, deverá estar ligada para todos os períodos entre *t* e o estado *ligado* seguinte.

A segunda estratégia analisa todas as unidades sobre o horizonte de planeamento. Se uma dada unidade está desligada durante um número pequeno de períodos, é considerado um cenário alternativo com aquela unidade ao longo do horizonte de planeamento.

A última estratégia procura unidades que só estão ligadas nas horas de ponta. Se essas unidades existirem é definido um novo cenário em que essa unidade é substituída por outras unidades que estão desligadas durante esses períodos mas que estão ligadas antes e/ou depois. Todas as estratégias de pesquisa ambicionam reduzir os custos de transição.

D. Detalhes de implementação

Uma das características mais atractivas das metaheurísticas é o facto de elas tenderem a ser independentes do problema, i.e., o núcleo principal da sua estrutura é o mesmo, independentemente do problema considerado. Por outro lado, o seu desempenho tende a ser altamente dependente das estratégias de pesquisa desenvolvidas para cada problema particular e, a maior parte das vezes, são desenhados diferentes módulos de software, codificando diferentes estratégias de pesquisa.

Em [20], diferentes foram desenvolvidas metaheurísticas em Visual C++ tendo sido consideradas três grandes objectos tiveram de ser considerados: 1) o problema, 2) a

vizinhança e 3) as heurísticas de pesquisa como está ilustrado na Figura 3.2. A *Master Class* do problema guarda todos os dados relacionados com o problema, como por exemplo as exigências da carga e da reserva girante, as características dos geradores e outros parâmetros, tal como o horizonte de planeamento.

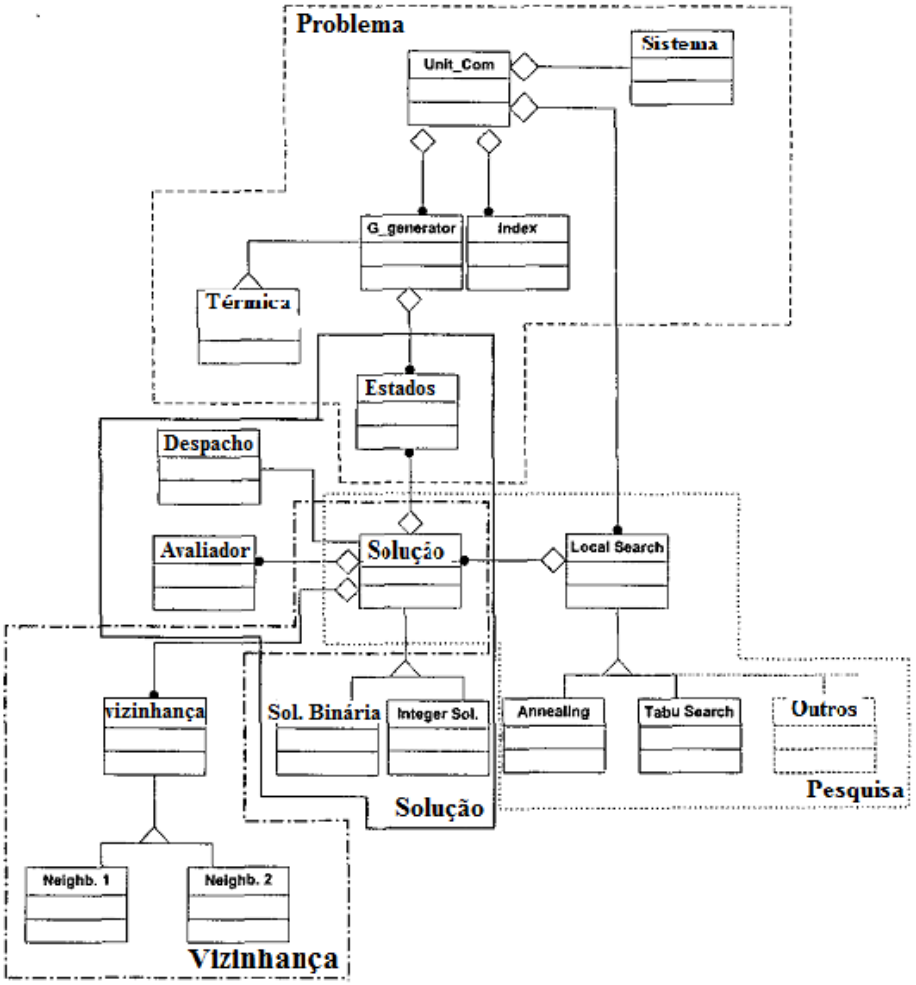

Figura 3.2 - Diagrama estático UML [20].

Para cada período de tempo, cada gerador pode ser desligado ou ligado, e se for ligado pode ter diferentes níveis de produção. Estes são os aspectos que descrevem os estados de cada gerador.

Os estados de todos os geradores ao longo de todo horizonte de planeamento representam uma solução para diferentes operações de vizinhança e podem ser aplicadas diferentes estratégias de pesquisa baseadas em métodos de pesquisa local. Depois de ser criada a estrutura de vizinhança, é utilizado um algoritmo capaz de calcular o nível óptimo de produção de cada unidade, algoritmo de despacho, e a qualidade da nova solução é avaliada por uma função de avaliação que reflecte o custo de produção associado.

E. Testes e resultados

De forma a validar a abordagem e o software desenvolvido em [20], foram realizados diversos testes computacionais, tendo em conta sistemas produtores com 10 e 20 unidades de produção. O problema com 20 unidades é baseado no problema de 10 unidades, considerando 10 unidades extra com as mesmas características das primeiras 10 e duplicando a carga e exigências de reserva girante. Em ambos os casos, a reserva girante foi considerada como correspondendo a 10% da carga. Os dados para o problema com 10 geradores estão ilustrados na Tabela 3.1.

Tabela 3.1 – Dados para o problema com 10 geradores [20]

	Unidade 1	**Unidade 2**	**Unidade 3**	**Unidade 4**	**Unidade 5**
Pmáx (MW)	455	455	130	130	162
Pmin (MW)	150	150	20	20	25
a ($h)	1000	970	700	680	450
b ($MWh)	16.19	17.26	16.60	16.50	19.70
c ($MW^{2}h)	0.00048	0.00031	0.002	0.00211	0.00398
Min Ligada (h)	8	8	5	5	6
Min Desligada (h)	8	8	5	5	6
Custo hot-start ($)	4500	5000	550	560	900
Custo cold-start ($)	9000	10000	1100	1120	1800
Horas cold- start(h)	5	5	4	4	4
Estado inicial (h)	8	8	-5	-5	-6
	Unidade 1	**Unidade 2**	**Unidade 3**	**Unidade 4**	**Unidade 5**
Pmáx (MW)	80	85	55	55	55
Pmin (MW)	20	25	10	10	10
a ($h)	370	480	660	665	670
b ($MWh)	22.26	27.74	25.92	27.27	27.79
c ($MW^{2}h)	0.00712	0.00079	0.00413	0.00222	0.00173
Min Ligada (h)	3	3	1	1	1
Min Desligada (h)	3	3	1	1	1
Custo hot-start ($)	170	260	30	30	30
Custo cold-start ($)	340	520	60	60	60
Horas cold- start(h)	2	2	0	0	0
Estado inicial (h)	-3	-3	-1	-1	-1

Neste artigo [20], o problema do *Unit Commitment* foi apresentado e resolvido com a abordagem do *Simulated Annealing*.

3.1.2 Segundo exemplo

A. Aspectos Gerais

Em [21] é proposta uma resolução do problema de *Unit Commitment* usando o *Simulated Annealing* e considerando rampas de subida ou descida de potência.

Por analogia, o algoritmo SA é usado para procurar a solução óptima (ou próxima da óptima) no problema combinatório do UC. Cada configuração do sistema físico corresponde à solução actual do problema de optimização, a energia dos átomos tem analogia com o valor da função objectivo e o último estado fundamental é equivalente ao mínimo global da função custo. Mais ainda, a temperatura desempenha o papel de parâmetro de controlo do processo de optimização.

Os passos básicos do algoritmo SA para este problema são os seguintes:

1. Encontrar aleatoriamente uma solução possível inicial que é considerada como solução actual X_i, e realizar o despacho económico de forma a calcular os custos de operação do sistema. Calcular os custos de ligação (*start-up costs*) através das expressões 3.1 e 3.2.

   Custo de ligação da unidade *i* na hora *t*:

$$STRT_{i,t} = b_{0,i}(1 - e^{\frac{\max(0,-Toff_{i,t-1})}{k_i}}) + b_{1,i} \tag{3.1}$$

   Nesta expressão, $b_{0,i}$, $b_{1,i}$, k_i são os coeficientes dos custos de ligação da unidade *i* e $Toff_{i,t}$ é um inteiro negativo indicando o tempo consecutivo que a unidade *i* está no estado desligada até à hora *t*. Esta função pode ainda ser definida da forma:

$$STRT_{i,t} = \begin{cases} S_{h,i}, se - Toff_{i,t-1} \le t_{cold,i} \\ S_{c,i}, senão \end{cases} \tag{3.2}$$

Nesta expressão $t_{cold,i}$ é o número de horas que a caldeira da unidade *i* demora a arrefecer, enquanto $S_{h,i}$ e $S_{c,i}$ são os custos de ligação da unidade *i* para uma ligação a quente e a frio, respectivamente.

O custo total é calculado através da expressão(3.3).

$$TC = \sum_{t=1}^{T} \sum_{i=1}^{NG} [ST_{i,t} FC_{i,t}(P_{i,t}) + ST_{i,t}(1 - ST_{i,t-1}) STRT_{i,t}] \qquad (3.3)$$

onde *TC* é o custo total do sistema, *T* é o período de despacho considerado, *NG* o número de unidades que estão a gerar e $ST_{i,t}$ representa o estado da unidade *i* na hora *t* (1 se a unidade está ligada, 0 se a unidade estiver desligada).

2. Determinar o valor inicial do parâmetro de controlo da temperatura $Temp = Temp_0$.
3. Inicializar o contador de iterações em $k=1$.
4. Procurar uma solução vizinha X_j através de uma perturbação aleatória na solução actual e calcular o novo custo total TC_j.
5. Se $TC_j \leq TC_i$, aceitar solução nova, colocar $X_i=X_j$ e $TC_i=TC_j$ e passar para o passo 6. Se $TC_j>TC_i$, calcular o desvio $\Delta C = TC_j - TC_i$ e gera um número aleatório de uma distribuição uniforme $U(0,1)$. Se $exp(-\Delta C / Temp) \geq U(0,1)$, aceitar a nova solução X_j substituindo X_i como solução actual.
6. Se *k* é menor que o comprimento da cadeia de *Markov*, incrementar o contador de iterações $k= k+1$ e voltar ao passo 4. Senão seguir para o passo 7.
7. Realizar a optimização local na vizinhança da última solução aceite.
8. Se o critério de paragem não for satisfeito, diminuir a temperatura *Temp* e voltar ao passo 3.

O algoritmo SA proposto adopta três diferentes mecanismos para a geração das soluções candidatas. O primeiro mecanismo muda aleatoriamente o estado das unidades durante apenas uma hora, enquanto os outros mecanismos mudam aleatoriamente o estado de uma unidade seleccionada durante um número aleatório de horas consecutivas. A combinação destes mecanismos resulta numa exploração bi-dimensional do espaço de soluções, acelerando a convergência do algoritmo. Por outro lado, estes mecanismos permitem que só sejam analisadas soluções admissíveis durante a execução do programa.

O valor inicial do parâmetro de controlo da temperatura é determinado de tal forma que na fase inicial do algoritmo, uma grande percentagem de soluções sejam aceites (por exemplo 85%).

O comprimento de cada cadeia de Markov do algoritmo SA é determinado como o mínimo valor entre um limite superior e um número de variáveis (produto do número de horas pelo número de unidades). Assim que a cadeia de Markov esteja completa, o parâmetro de controlo da temperatura é diminuído considerando um factor de arrefecimento próximo da unidade, de modo a simular um arrefecimento lento.

$$Temp_{k+1} = \beta . Temp_k \tag{3.4}$$

Nesta expressão, $Temp_k$ é a temperatura no patamar k e β é uma inferior a 1, mas próxima da unidade.

Para acelerar a convergência do algoritmo SA, é implementado um método local de optimização para cada iteração. Este método identifica a melhor solução na vizinhança da última solução aceite usando mecanismos iterativos desenvolvidos para gerar novas soluções de forma aleatória.

O algoritmo SA termina quando um dos critérios seguintes for satisfeito: um número pré-definido de cadeias de *Markov* tiver sido executado sem aceitar nenhuma solução nova ou ser atingida uma temperatura pré-seleccionada.

B. Visão geral do algoritmo proposto

No caso das restrições das rampas não serem tidas em conta, o despacho económico para cada solução fornecida pelo SA pode ser independentemente realizado para todas as horas. De forma a incluir taxas de rampa no procedimento referido, é implementado um despacho dinâmico através da realização quer de uma sequência em sentido inverso quer em sentido dianteiro de despachos económicos horários convencionais com limites dos geradores convenientemente modificados.

Durante a execução do algoritmo SA, é gerada uma nova solução candidata X_{nova} com uma pequena perturbação em relação à solução corrente $X_{corrente}$ mudando o estado de um número aleatório de unidades durante o período de selecção aleatório de horas consecutivas H_i, onde $H_i \in [H_1, H_2]$. Assumindo que a solução corrente $X_{corrente}$ é viável no que diz respeito às restrições de taxa de rampa, o algoritmo proposto aplica dois procedimentos alternativos: despacho dinâmico no sentido dianteiro e outro no sentido inverso. Em ambos os casos, é realizado um despacho convencional independente para cada hora considerada usando limites dos geradores modificados. Estes procedimentos garantem que as rampas são tidas em consideração para a solução do sub-problema do despacho económico para a nova solução X_{nova}.

C. *Forward Dynamic Dispactch*, FDD

O FDD é baseado na solução dinâmica de um sub-problema de despacho económico para cada hora *t* do intervalo *[H_1, T]*.Os limites modificados dos geradores a serem usados no FDD são definidos de acordo com (3.5) e (3.6).

$$Pmod\_min_{i,t} = MAX(Pmin_i, P_{i,t-1} - RATEDN_i) \quad (3.5)$$

$$Pmod\_max_{i,t} = MIN(Pmax_i, P_{i,t-1} + RATEUP_i) \quad (3.6)$$

Nestas expressões, $Pmod\_min_{i,t}$ e $Pmod\_max_{i,t}$ representam os limites mínimo e máximo modificados dos limites de uma unidade *i* à hora *t*, *RATEDNi* e *RATEUPi* representam respectivamente a máxima diminuição permitida e máximo aumento permitido do valor de produção da unidade *i* numa determinada hora. Com base na análise acima, os passos básicos do FDD são:

1. Colocar contador $t=H_1$.
2. Calcular os limites modificados das unidades na hora *t*, usando as expressões (3.5) e (3.6).
3. Realizar o despacho económico para a hora *t* de forma a calcular a potência produzida $P_{i,t}$ de cada unidade *i* à hora *t*.
4. Se $t<T$, aumentar o contador $t=t+1$ e voltar ao passo 2.

D. *Backward Dynamic Dispatch*,BDD

O BDD é baseado na solução dinâmica do sub-problema do despacho económico para cada hora *t* do intervalo [1, H_2]. Neste caso, os limites modificados dos geradores são definidos por (3.7) e (3.8).

$$Pmod\_min_{i,t} = MAX(Pmin_i, P_{i,t+1} - RATEDN_i) \quad (3.7)$$

$$Pmod\_max_{i,t} = MIN(Pmax_i, P_{i,t+1} + RATEUP_i) \quad (3.8)$$

A aplicação do BDD é similar à do FDD, mas a direcção do despacho dinâmico é reservada. Consequentemente, os passos básicos da BDD são:

1. Colocar contador de tempo $t= H_2$.
2. Calcular os limites modificados das unidades à hora t, usando (3.7) e (3.8).

3. Realizar o despacho económico para a hora t de forma a calcular a potência produzida por cada unidade.
4. Se $t>1$, diminuir o contador de tempo $t= t-1$ e voltar ao passo 2.

E. Mecanismo de Selecção

A maior contribuição do método proposto comparando com o despacho económico tradicional consiste na incorporação do mecanismo de selecção, que determina o tipo de procedimento dinâmico (FDD ou BDD) que deve ser realizado para cada nova solução examinada pelo algoritmo SA. A ideia principal é seleccionar o procedimento que leva a menos execuções do sub-problema do despacho económico. Este mecanismo de selecção contribui para uma redução do tempo de cálculo.

Como já foi mencionado, o FDD é aplicado para cada hora de um intervalo [H_1, T] e, por esse motivo, origina N_F execuções do sub-problema do despacho económico, onde:

$$N_F = T-H_1+1 \tag{3.9}$$

Similarmente, a BDD é aplicada para cada hora no intervalo [1, H_2] e, por isso, requer H_2 execuções do sub-problema do despacho económico.

Na mesma base da análise acima, o procedimento apropriado de despacho para cada caso é determinado comparando valores de N_F *e* de H_2 de tal modo que:

- Se $N_F<H_2$, é realizado o FDD;
- Se $N_F>H_2$, é realizado o BDD;
- Se $N_F=H_2$, o procedimento é aleatoriamente seleccionado.

F. Avaliação do método proposto

Como foi já mencionado, a aplicação do método proposto é baseada no pressuposto que a nova solução candidata X_{nova} é gerada com uma pequena perturbação em relação à solução corrente $X_{corrente}$, que é viável no que diz respeito às restrições das rampas. No entanto, não há garantia que a solução inicial do algoritmo SA seja viável no que diz respeito a essas restrições. Por isso, durante as primeiras iterações do algoritmo, as restrições das rampas podem ser violadas por um número de horas. Desta forma, é conseguida uma exploração mais extensa do espaço de soluções. No entanto, à medida que o algoritmo prossegue, estas violações vão sendo eliminadas pela aplicação repetida do método dinâmico proposto e, assim, é rapidamente encontrada uma solução viável. Em seguida, só soluções viáveis são examinadas pelo algoritmo SA. Além disso, já que a potência produzida não é coordenada por

um período inteiro de despacho, a solução final é *subóptima*. No entanto, o método proposto garante que são encontradas soluções próximas do óptimo em tempos razoáveis.

G. Resultados numéricos

A eficácia do algoritmo descrito foi testada num sistema com dez unidades de produção, considerando um período de despacho de 24 horas. De forma a mostrar a eficiência do modelo proposto na resolução de problemas reais de *Unit Commitment*, os dados do sistema de teste foram ainda escalados apropriadamente com o objectivo de obter um novo sistema com 40 unidades de produção.

De forma a ilustrar o impacto das restrições das rampas, foram examinados dois casos de estudo. O objectivo do primeiro caso de estudo foi provar a eficiência do algoritmo proposto em resolver um problema de UC reais, sem ter em consideração as restrições das taxas das rampas. No segundo caso de estudo, as restrições associadas às rampas foram incluídas usando o método dinâmico proposto, enquanto os dados do outro sistema se mantiverem os mesmos. Os dados consideram as restrições de rampas para cada unidade na informação relativa ao segundo caso de estudo. Sem perda de generalidade, foi assumido que o valor do RATEUP de uma unidade é igual ao valor da RATEDN. Em conformidade, a rampa de cada unidade de produção foi assumida igual a 20% da sua potência máxima, como se indica mostrado na Tabela 3.2.

Tabela 3.2 - Rampa das unidades de produção

Nº da unidade	Rampa (MW/h)	Nº da unidade	Rampa (MW/h)
1	90.0	6	16.0
2	90.0	7	17.0
3	26.0	8	11.0
4	26.0	9	11.0
5	32.4	10	11.0

Devido à natureza aleatória do algoritmo SA, o programa foi executado 10 vezes para cada caso de estudo, de forma a validar a estabilidade do algoritmo. A temperatura inicial para cada situação foi determinada para cada caso de estudo usando o procedimento descrito no artigo correspondente. Em todos os estudos, foi usado o processo de arrefecimento, com β=0,95. Para todos os casos foi adoptado um critério de paragem correspondente à temperatura final de 0,001 ou à não-aceitação de uma solução durante duas cadeias de Markov consecutivas. O comprimento de cada cadeia de Markov foi determinado como o mínimo valor entre 800 e o número de variáveis do problema. Finalmente, foram usadas 50 iterações para cada pesquisa local em cada nível de temperatura.

Os resultados da simulação do primeiro caso de estudo são apresentados na Tabela 3.3. Esta tabela inclui o melhor custo, o pior custo e custo médio obtidos através de dez execuções do algoritmo proposto. Para além disso, a Tabela 3.3 mostra a comparação entre o algoritmo SA e três outros algoritmos para o sistema de 40 unidades de produção. Os resultados foram obtidos usando o método da Relaxação Lagrangiana (LR), algoritmos genéticos (GA) e *Seeded Memetic* (SM). A comparação com estes métodos mostra que o algoritmo SA fornece resultados de alta qualidade. Por outro lado, a pequena variação entre o custo da melhor e da pior solução comprova a robustez do algoritmo desenvolvido.

A Tabela 3.3 apresenta ainda a média de tempo de execução do SA para o caso de estudo examinado. Dado que o tempo de execução é dependente do computador utilizado, o tempo de cálculo dos outros métodos não são mencionados. De qualquer forma, pode-se verificar que as exigências de tempo do modelo proposto são razoáveis.

Tabela 3.3 - Comparação de resultados para o sistema de 40 unidades do primeiro caso de estudo.

Método	Melhor Custo ($)	Custo Médio ($)	Pior Custo($)	Tempo médio (sec)
SA	2250063	2252125	2254539	88.28
LR	2250223	2250223	2250223	-
GA	2252909	2262585	2269282	-
SM	2249589	2249589	2249589	-

A Tabela 3.4 mostra os resultados da simulação para o segundo caso de estudo do sistema de 40 unidades de produção. Os parâmetros de controlo do algoritmo SA foram determinados da mesma forma que o primeiro caso de estudo, e as restrições das rampas foram incluídas. Pode-se ver que, como esperado, a inclusão das restrições leva a um ligeiro aumento do custo total do sistema. O tempo de execução médio do algoritmo é também ele aumentado, mas mantém-se em valores aceitáveis. Todavia, as exigências de tempo são significantemente maiores se o despacho económico for simultaneamente realizado para o período inteiro de despacho.

Tabela 3.4 - Comparação de resultados para o sistema de 40 unidades dos casos de estudo 1 e 2.

Caso de estudo	Melhor Custo ($)	Custo médio ($)	Pior Custo ($)	Tempo médio (sec)
1	2250063	2252125	2254539	88.28
2	2255864	2256971	2258897	199.55

3.2 Planeamento de expansão a longo prazo do sistema de transmissão

Outro exemplo interessante é o descrito em [22]. Nesta publicação pretende-se efectuar o planeamento a longo prazo do sistema de transmissão, utilizando um algoritmo com uma abordagem usando o *Simulated Annealing* e tendo em consideração os preços marginais de longo prazo. Em seguida será exposto o algoritmo do plano de expansão da rede de transmissão, nos seus vários passos.

3.2.1 Formulação

Para ter em conta a natureza discreta do problema, o planeador prepara uma lista de possíveis componentes para reforçar ou construir. Cada elemento desta lista tem um custo associado e será eventualmente seleccionado pelo algoritmo de forma a integrar o plano de expansão num ano específico. O custo de investimento do plano, *IC*, corresponde à soma dos custos de investimento, IC_p, ao longo dos *np* períodos no horizonte, ajustados adequadamente usando uma taxa de actualização *r*.

$$IC = \sum_{p=0}^{np-1} \frac{IC_p}{(1+r)^p} \tag{3.10}$$

Os custos de operação são determinados resolvendo um problema do despacho curto prazo para cada ano considerado no horizonte de planeamento. Se este admitir que são seleccionados um número de elementos de uma lista de reforços e expansões distribuído ao longo do horizonte de planeamento, deve-se resolver tantos problemas de despacho quantos *np* períodos forem considerados. Cada um inclui reforços e expansões em todos os períodos anteriores. O custo de operação total é calculado usando uma expressão similar à (3.10) para referir os custos de operação ao ano inicial. Finalmente, é calculada a energia não fornecida esperada para todos os períodos usando uma simulação de *Monte Carlo* pseudo-cronológica. Assim, cada plano é caracterizado pelos custos operacionais, *OC*, custos de investimento, *IC*, e energia esperada não fornecida, *EENS*, pelo que é formulado um problema multicritério.

3.2.2 Problema multicritério

Há várias abordagens para lidar com problemas multicritério. Elas incluem a determinação de soluções não dominadas seguindo-se uma abordagem de decisão que pode

integrar uma análise de *trade-off* e métodos interactivos que partem de uma solução não dominada que é melhorada sucessivamente de acordo informação suplementar fornecida pelo agente de decisão.

Neste caso, foi adoptado um método interactivo para resolver um problema genérico tal como (3.11) a (3.15).

$$\min f = [OC, IC, EENS] \quad (3.11)$$

Suj: $A.X + B.Y \leq d$ (3.12)

$$C.X \leq c \quad (3.13)$$

Limites de Y (3.14)

$$X \in \{x_1, x_2, \dots, x_n\} \quad (3.15)$$

Nesta formulação, f é o vector de critérios, Y são as variáveis de operação de curto prazo, X representa uma lista discreta de reforços e expansões, (3.12) indica as restrições de operação para cada período do horizonte de planeamento e (3.13) representa os limites do número de reforços ou expansões por período ou limites relacionados com a quantia de que se dispõe para investir num ano.

Uma solução não dominada deste problema pode ser obtida usando o método das restrições detalhado em [23]. Neste método todos os objectivos excepto um são convertidos em restrições usando níveis de aspiração levando a um problema com um único objectivo, podendo o agente de decisão definir os níveis de aspiração para os custos de investimento e *EENS*.

3.2.3 Identificação de soluções não dominadas

O problema anterior com um único objectivo pode ser resolvido usando o *Simulated Annealing* usando uma lista de expansões e reforços a partir da qual o algoritmo selecciona componentes a construir e o respectivo ano. O algoritmo organiza elementos de uma lista de expansões e reforços num plano dinâmico, de acordo com as indicações seguintes.

i) Considerar o actual sistema transmissão/produção como a topologia inicial e correspondente à solução x^0 ;

ii) Analisar a solução actual:

   a. Calcular os custos de investimento, *IC*, e a *EENS*;

   b. Calcular preços marginais de curto prazo (descrito em pormenor em [22]) para a topologia corrente;

c. Construir a função de avaliação EF^0 como a soma dos *OC* e penalizações para *IC* e *EENS* no caso de estarem fora dos limites especificados pelo agente de decisão;
d. Atribuir EF^0 a EF^{optimo} e a $EF^{corrente}$;
e. Atribuir x^0 a x^{optimo} e a $x^{corrente}$;
f. Colocar o contador de iterações, *ITC*, em 1;

Colocar o contador de soluções piores, *WSC*, em zero;

iii) Identificar um novo plano x^{novo} na vizinhança do plano corrente. Para isto sortear um dos períodos no plano de horizonte e, em seguida, depois sortear uma nova instalação a construir, de entre aquelas de uma lista de possíveis adições, ou desactivações de instalações já existentes. Uma nova instalação estará, então, activa nos períodos subsequentes;

iv) Analisar o novo plano:

a. Verificar se o número de instalações a construir por período excede o limite especificado. Se sim, a solução é descartada regressando-se a iii);

b. Calcular OC^{nova}, IC^{nova} e $EENS^{nova}$ e obter o novo valor da função de avaliação, EF^{nova};

v) Se $EF^{nova} < EF^{optima}$ então:

a. Atribuir EF^{nova} *a* EF^{otpima} *e a* $EF^{corrente}$;
b. Atribuir x^{nova} *a* x^{optima} *e a* $x^{corrente}$;
c. Colocar o contador de piores soluções, *WSC*, a zero;

vi) Se $EF^{nova} \geq EF^{optima}$ então:

a. Sortear um número aleatório $p \in [0.0;1.0]$
b. Calcular a probabilidade de aceitar piores soluções $p(x^{nova})$ através de (3.16);

$$p(x^{nova}) = e^{\frac{EF^{corrente} - EF^{nova}}{K.T}} \tag{3.16}$$

c. Se $p \leq p(x^{nova})$ então atribuir x^{nova} a $x^{corrente}$ e EF^{nova} a $EF^{corrente}$;
d. Aumentar o contador de piores soluções, *WSC*, de uma unidade.

vii) Se *WSC* é maior que um número especificado de iterações sem melhoria, então avançar para ix);

viii) Se o contador de iterações *ITC* é maior que o máximo número de iterações por nível de temperatura então:

a. Diminuir o nível de temperatura corrente de factor de arrefecimento β;
b. Se o novo nível de temperatura é mais pequeno que o nível mínimo permitido então avançar para ix);
c. Colocar o contador de iterações, *ITC*, em 1;
   Senão, incrementar em uma unidade o contador *ITC*;
   Voltar para iii);

ix) Fim.

3.2.4 Resultados Obtidos

A formulação do problema de expansão de uma rede de transmissão foi aplicada a diversos casos de estudo, nomeadamente à Rede Nacional de Transporte portuguesa. Foi considerada como ponto de partida a topologia da rede existente em 2004 e foi realizado o planeamento da expansão num horizonte considerando 6 anos. O plano construído revelou-se de boa qualidade, nomeadamente tendo em conta os custos de operação, de investimento e o valor anual esperado da energia não fornecida [22].

3.3 Optimização da Iluminância em ambientes fechados

Será agora apresentado um outro trabalho em que é usada a técnica *Simulated Annealing* para optimizar a Iluminância num ambiente fechado. Este trabalho está descrito ao pormenor em [24].

3.3.1 Considerações introdutórias do problema

O objectivo de um projecto de iluminação é garantir um nível constante de luminância numa determinada superfície. A uniformidade da Iluminância pode ser caracterizada por uma das três seguintes relações:

$$\frac{E_{\min}}{E_H};\ \frac{E_H}{E_{\max}};\ \frac{E_{\min}}{E_{\max}} \tag{3.17}$$

Nestas expressões:

E_H: iluminância média nos pontos da superfície

E_{max}: máxima iluminância nos pontos da superfície

E_{min}: mínima iluminância nos pontos da superfície

De acordo com [25], estas relações não devem assumir valores inferiores a 0.33, e idealmente devem ser próximas de 0.7. O nível recomendado de iluminância média num ambiente depende de vários factores, incluindo: tipo, velocidade e precisão da actividade realizada nesse ambiente, bem como a idade das pessoas que usam frequentamente esse ambiente e regulações específicas do país ou região.

O método lúmen é um dos tipicamente utilizados para especificar o número de luminárias e a sua distribuição, de modo a obter a iluminância média especificada. Neste método o número de luminárias é calculado da seguinte forma:

$$n_L = \frac{\varphi}{\varphi_L} \tag{3.18}$$

Nesta expressão:

φ : fluxo luminoso total

φ_L : fluxo luminoso de cada luminária

O fluxo total é obtido através da expressão (3.19).

$$\varphi = \frac{E_H . S}{C_{Ut} . C_{Man}} \tag{3.19}$$

Nesta expressão:

C_{Ut}: coeficiente de utilização

C_{Man}: coeficiente de manutenção

S: área de superfície

O método lúmen propõe luminárias com a mesma potência eléctrica desde que o valor de φ_L obtido por (3.18) seja diferente para todas as luminárias, tal como a sua distribuição simétrica. Estas duas características do método lúmen não garantem valores óptimos para a iluminância nomeadamente em termos da sua uniformidade. Além do mais, este método despreza diversos parâmetros relevantes, tais como custos de instalação e custos operacionais. Deve ser sublinhado que custos operacionais baixos são essenciais para garantir uma boa eficiência energética.

Em [24] é proposto um método para planear a instalação de iluminação de ambientes fechados, optimizando parâmetros tais como o nível de iluminância e uniformidade, custos de

instalação e operação usando o *Simulated Annealing*. Os resultados sugerem que um projecto óptimo exige o uso de diferentes potências e uma distribuição não simétrica de luminárias. Além disso, os valores obtidos para os parâmetros anteriormente referidos foram melhores que aqueles obtidos com o método lúmen descrito em [25].

3.3.2 Métodos

A característica mais relevante do algoritmo descrito em [24] é a utilização do *Simulated Annealing* para optimizar o projecto de iluminação.

Considerando o que já foi referido no Capítulo 2 sobre esta metaheurística, é apresentado na Figura 3.4 o algoritmo adoptado em [24] para resolver este problema.

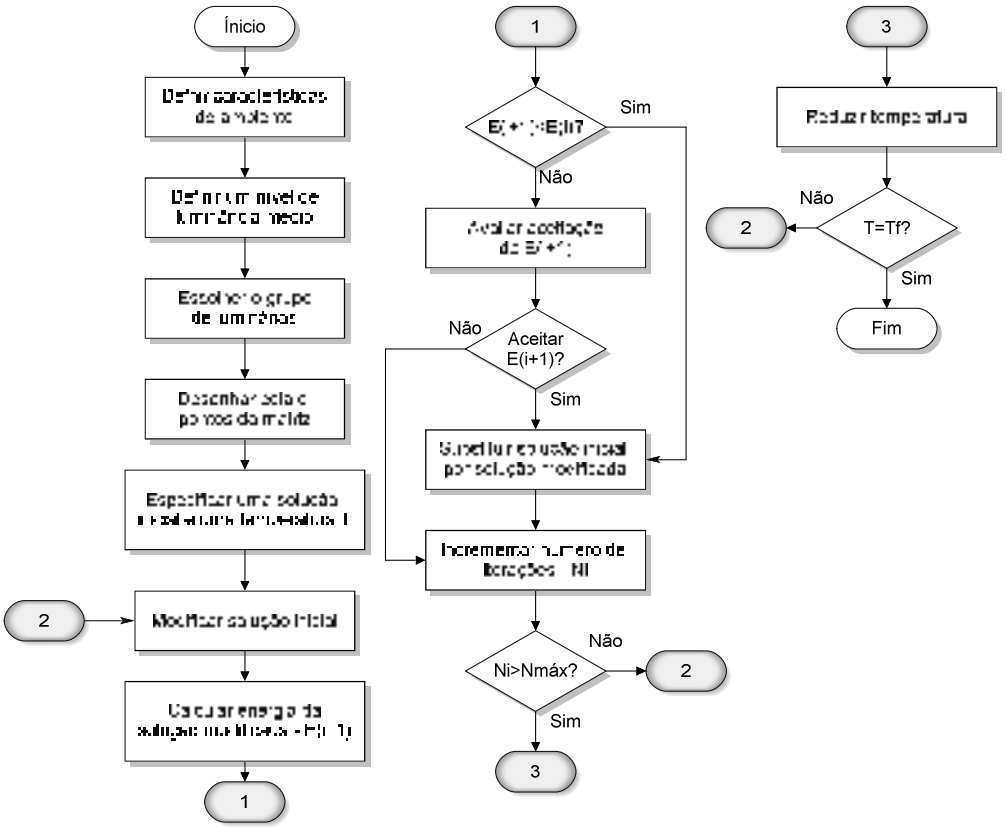

Figura 3.3 - Fluxograma do método proposto para optimizar projecto de iluminação [23].

3.3.3 Função de Avaliação das soluções

A função de avaliação das soluções proposta é composta por três termos indicados em (3.20).

$$E = \mu.CI + \delta.CO + \theta..DP \tag{3.20}$$

Nesta expressão:

CI: custo de instalação;

$$CI = \sum_i L_i P_i \tag{3.21}$$

L_i: número da luminária do tipo *i*;

P_i: custo da luminária de tipo *i*;

CO: custo de operação dado por (3.22);

$$CO = C.\sum_i L_i.Pt_i \tag{3.22}$$

C:custo do kWh;

Pt_i: potência da luminária do tipo *i*;

DP: desvio padrão da iluminância dado por (3.23)

$$DP = \sqrt{\sum_{i=1}^{N} \frac{(NI_i - ND)^2}{N}} \tag{3.23}$$

N: número de pontos na matriz do plano de trabalho;

NI_i: nível de iluminância no ponto i do plano de trabalho;

ND: nível de iluminância desejado.

Na expressão (3.20), o termo DP permite que o *Simulated Annealing* optimize o nível de iluminância e a sua uniformidade. Os custos operacionais e de instalação são tidos em conta considerando os coeficientes μ, δ e θ. Uma vez que a distribuição das luminárias fique definida, o valor do nível da iluminância NI para cada ponto da matriz que representa a área a iluminar é calculado, usando o método ponto a ponto descrito em [25]. Neste cálculo, são consideradas a iluminação directa e indirecta provenientes dos tectos.

3.3.4 Parâmetros do *Simulated Annealing*

O processo de arrefecimento é caracterizado pela expressão (3.24).

$$T_{i+1} = \beta.T_i \tag{3.24}$$

Nesta expressão β está compreendida entre 0 e 1 e corresponde ao coeficiente de arrefecimento da temperatura. A temperatura final do *Simulated Annealing* foi estabelecida em 1.

O ambiente usado para ajustar os parâmetros e testar o método proposto foi uma sala de uma escola pública. As suas características fazem com que o nível de iluminância média tenha sido estabelecido em 300 lux.

3.3.5 Resultados

De forma a ajustar os coeficientes μ, δ e θ da função de avaliação (3.20), foi usada uma solução típica para o ambiente testado. Foram obtidos valores de CO=US$26.47, CI=US$682.35 e DP=110 lux para os termos de (3.20). Com o objectivo de manter uma contribuição igual por parte dos termos da equação de energia, os valores de μ, δ e θ foram colocados em 28.3, 1 e 10.5 respectivamente. Para estes valores, a optimização da função de avaliação não confere prioridade a nenhum destes termos. Quando foi usado um processo empírico com o objectivo de dar prioridade ao custo de operação e à uniformidade de iluminância, os valores para μ, δ e θ foram 75, 1 e 220 respectivamente.

Depois de alguns testes efectuados, foram obtidos os valores a usar para a temperatura inicial T_i, o coeficiente de arrefecimento β e o número de iterações ao fim do qual se reduz a temperatura *k*. Esses valores foram fixados em 5000, 0.99 e 10 respectivamente.

Os resultados obtidos usando este método permitiram obter valores de uniformidade de iluminância de 0.6 no plano de trabalho. Estes valores são em muito superiores aos obtidos pelo método lúmen. Os autores de [24] referem que estes resultados são de muito melhor qualidade quando comparados quando obtidos com o método lúmen detalhado em [25].

3.4 Comentários

Através dos quatro exemplos demonstrados usando o *Simulated Annealing*, ficou mostrado o interesse do uso desta técnica. Em problemas tão distintos, embora da mesma área, esta técnica revelou ter grande flexibilidade adaptando-se ao problema em questão. Foi ainda evidenciado que esta técnica conduz em geral a bons resultados, testemunhando por isso uma das razões para ter sido escolhida como técnica a utilizar neste trabalho.

Capítulo 4

Descrição do problema

Neste capítulo será exposta a metodologia implementada. Será explicada a necessidade da ferramenta criada, as considerações adoptadas no seu desenvolvimento, as suas capacidades e a descrição da forma como foi implementada. O objectivo consistia em criar uma metodologia que fosse capaz de minimizar o erro entre um histórico de valores, tipicamente de energia consumida horária, e um diagrama de cinco patamares. A forma como foi colocado este problema está relacionada com o facto de se pretender que esta ferramenta interaja com o modelo VALORAGUA, utilizado pela EDP Produção. De seguida será dada um breve esclarecimento sobre este modelo.

4.1 VALORAGUA

4.1.1 Introdução

O VALORAGUA é uma aplicação computacional de ajuda à decisão no planeamento do sistema eléctrico. Esta ferramenta tem especial ênfase no uso óptimo de um sistema misto, integrado em áreas de carga/produção ligadas por uma rede de transmissão. É considerada uma abordagem marginal levando ao cálculo dos preços nodais, reflectindo a variação da função custo se a carga no nó aumentar uma unidade [26].

Usualmente o horizonte total é um ano e o intervalo de tempo é um mês ou uma semana, geralmente chamado de período. Cada período é constituído por cinco sub períodos denominados de degraus de carga ou postos horários [26].

Esta ferramenta de cálculo fornece essencialmente a informação necessária para auxiliar o processo do decisor no sector eléctrico. Tal como acontece em muitos modelos de

optimização, é um resultado de uma série de actualizações, devido à evolução natural de técnicas matemáticas, de forma a cobrir novas restrições e condições de operação do sistema eléctrico de potência e também de forma a ser adaptada a um mercado de electricidade aberto [26].

O modelo permite um controlo detalhado do sistema eléctrico de energia, representado como uma interligação dos principais subsistemas: subsistema de produção, subsistema de transmissão e subsistema de consumo. Cada componente de todos os subsistemas é completamente individualizado, com a sua identificação e topologia de ligação à rede eléctrica e/ou rede hidráulica, e realiza uma actividade física ou económica.

4.1.2 Componentes do sistema eléctrico

A carga pode ser definida como carga fixa ou carga flexível. A carga fixa é modelada para cada período, mês ou semana, e de acordo com as respectivas variações de potência no período, representado por cinco degraus. Por outro lado, a carga flexível é modelada como uma relação preço/quantidade para cada período. Geralmente esta é realizada usando uma curva para o preço, sendo as quantidades ajustadas no mercado, dependendo dos preços de mercado. A carga flexível pode também representar contratos com os clientes [26].

Uma das características mais importantes deste modelo é a representação muito detalhada do sistema hidroeléctrico que tem um conjunto de redes hidráulicas, cada uma especificando uma cascata. Uma cascata consiste num conjunto de albufeiras ligadas por um ou mais componentes como centrais com turbinas, centrais com bombagem e descarregadores de cheias.

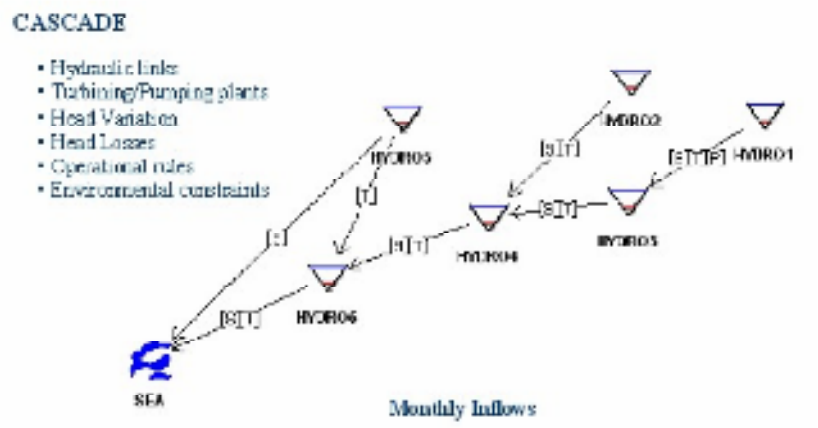

Figura 4.1- Ilustração de uma rede hidráulica.

O controlo detalhado da rede hidráulica permite a representação de:

- Cascatas hidroeléctricas, tendo em conta as ligações entre albufeiras e centrais hidroeléctricas;
- Centrais reversíveis (com turbinagem e bombagem) com a possibilidade de analisar semanalmente e sazonalmente o armazenamento para bombagem;
- Utilização da água para outros usos que não seja a produção de energia (consumo humano, fins de rega, fins ecológicos).

O subsistema térmico consiste num conjunto de centrais térmicas, cada uma composta por um número de unidades que, usando combustíveis apropriados, produzem electricidade que é entregue à rede. O custo associado de produção de energia por uma central térmica pode ser representado por uma função quadrática convexa da potência produzida. O modelo pode definir um programa de manutenção óptima para o subsistema de produção através de centrais térmicas [26].

A rede de transmissão eléctrica é modelada como uma rede orientada onde os nós representam as áreas geográficas, associadas a zonas de produção/consumo e as ligações representam o transporte ou linhas de interligação [26].

São consideradas no modelo, as perdas e a primeira lei de Kirkchof no sistema de transmissão. A operação da rede é modelizada pelo modelo DC sendo considerada uma estimativa das perdas em cada ramo de tal modo que metade do seu valor é adicionado à carga nas extremidades de cada ramo [26].

Quando se opera com sistemas mistos hidro-térmicos o maior desafio está relacionado com a gestão das albufeiras em parte devido à incerteza dos fluxos mas também devido ao limite de capacidade de armazenamento das albufeiras e a decisão de usar a água naquele momento ou de armazenar para uso futuro, dado que a decisão tenha implicações no presente e custos esperados futuros.

4.1.3 Aplicações

O modelo VALORAGUA fornece uma grande variedade de resultados para todos os componentes do sistema, para todos os intervalos de tempo e degraus de carga e para todas as séries hidrológicas. Usualmente são usados 40 anos para representar a hidrologia. Estão disponíveis resultados físicos, operacionais e económicos, que permitem várias análises.

Algumas aplicações interessantes do modelo VALORAGUA são:

- Optimização das principais características de um projecto hidroeléctrico;
- Avaliação económica de projectos hidroeléctricos;

- Delimitação de áreas interessantes para alocação de futuras centrais térmicas;
- Cálculo de custos de produção marginais para integrar em sistemas tarifários;
- Estudos probabilísticos sobre emissões atmosféricas possíveis e uso de quantidades de combustível;
- Gestão das centrais hídricas tendo em conta o mercado de electricidade.

4.2 Metodologia implementada

Como já foi referido, a ferramenta desenvolvida é capaz de minimizar o erro entre um histórico de valores horários e um diagrama de cinco patamares. Um diagrama deste género é necessário, uma vez que se pretende que esta ferramenta interaja com o modelo VALORAGUA, o qual, como já foi dito, opera com este tipo de formato. Esse diagrama vai sendo transformado ao longo das várias iterações do algoritmo, sendo alteradas a largura e altura dos respectivos patamares com o objectivo de alcançar o valor óptimo, ou seja aquele que minimiza o erro. Para resolver um problema de optimização deste género é aconselhável o uso de uma metaheurística, porque se trata de um problema altamente combinatório. Foi isso que foi feito neste trabalho, sendo escolhido o *Simulated Annealing* como metaheurística a utilizar. A Figura 4.2 mostra um exemplo de um diagrama deste género retirado directamente do software desenvolvido.

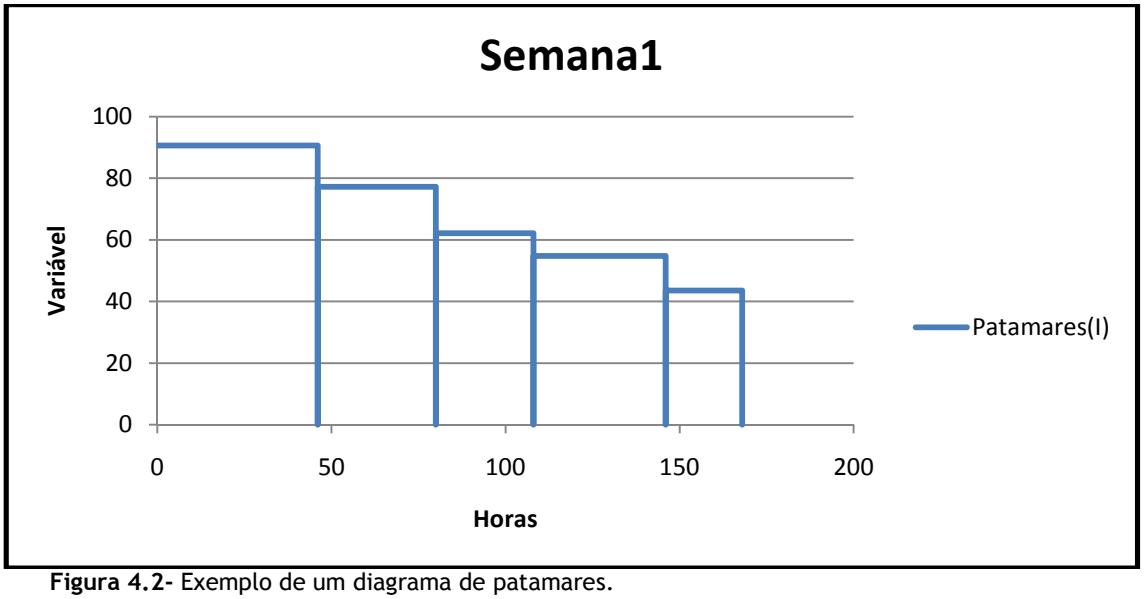

Figura 4.2- Exemplo de um diagrama de patamares.

Este exemplo refere-se a um diagrama de patamares que, depois de corrido o algoritmo, se ajusta a um histórico de valores de preços horários do mercado electricidade referentes a uma semana.

4.2.1 Descrição da ferramenta

A ferramenta que minimiza o erro entre um diagrama de patamares ajustável e um histórico de valores horários, tipicamente de energia consumida horária ou de preços horários de electricidade, foi implementada em ambiente *Visual Basic Applications* (VBA) e alcança os valores pretendidos através do uso do *Simulated Annealing*, referido em pormenor no Capítulo 2. Como se pode verificar pela Figura 4.2 pretende-se que a ferramenta crie um diagrama de patamares óptimo e que este seja apresentado com patamares sucessivos decrescentes. É também necessário que os valores das larguras dos patamares, ou seja o número de horas, sejam apresentados como valores inteiros, devido a exigências do modelo.

4.2.2 Ambiente de trabalho

A Figura 4.3 ilustra o ambiente de trabalho que é utilizado quando se trabalha com a ferramenta criada neste projecto.

Figura 4.3 - Excerto do ambiente de trabalho da ferramenta criada.

Do lado esquerdo existem alguns botões que servem para filtrar o problema desejado. Isto porque a optimização pretendida pode incidir sobre valores horários relativos a vários períodos de tempo.

Do lado direito aparecem os valores horários que surgem como entradas no algoritmo de optimização. Estes valores serão numa primeira fase ordenados de forma decrescente e

depois servirão para comparar o seu valor e o valor do patamar para calcular o erro existente e realizar a sua consequente minimização. De notar que é possível importar até cinco séries de dados diferentes e fazer uma optimização conjunta. Esta optimização é realizada com o pressuposto de que será optimizado o erro total, com a restrição de as larguras referentes ao mesmo patamar de todas as séries serão necessariamente iguais. A Figura 4.4 ilustra uma outra parte do ambiente de trabalho que surge quando o algoritmo está a ser executado e quando este converge. Como se pode verificar, quando há mais do que uma série de dados a serem importados são criados mais patamares a serem optimizados.

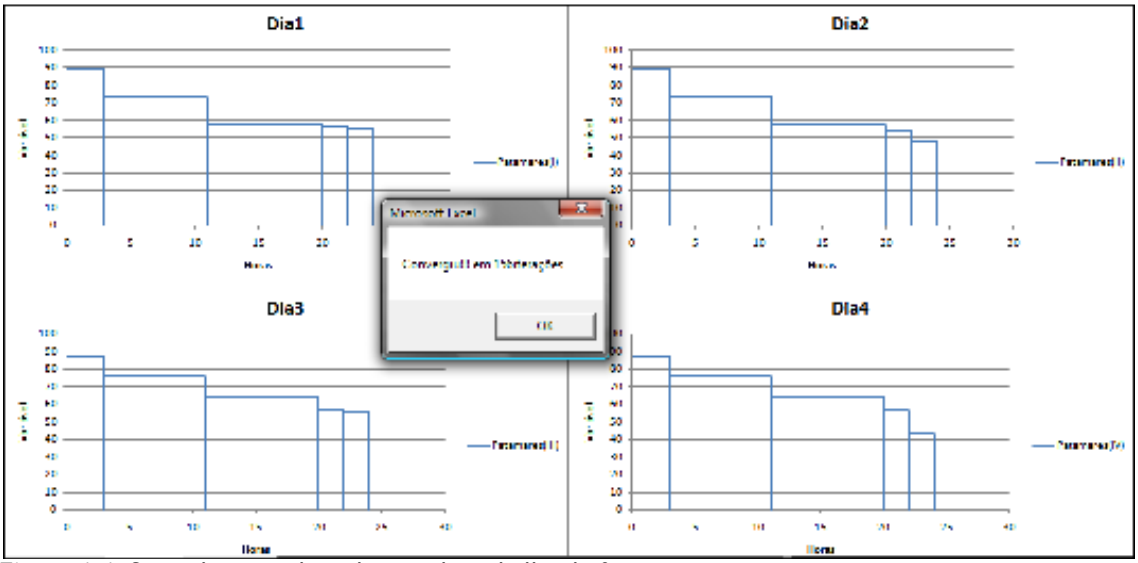

Figura 4.4- Segunda parte do ambiente de trabalho da ferramenta.

4.2.3 Solução inicial

Num método de optimização baseado numa metaheurística a solução inicial desempenha um papel importante. De notar que não sabendo à partida onde se encontram as boas soluções do problema, escolher uma solução inicial próxima ou distante da óptima pode condicionar os resultados.

No caso concreto deste problema não há uma indicação clara sobre uma boa solução inicial que traduza melhores resultados. Nesse sentido, foram estabelecidas as soluções iniciais indicadas na Tabela 4.1 como soluções iniciais por defeito. Mais adiante serão utilizadas outras soluções iniciais para realizar outras análises.

Tabela 4.1- Soluções iniciais - duração acumulada de cada patamar.

Período	L_i (Horas)				
	L_1	L_2	L_3	L_4	L_5
Dia	4	9	14	19	24
Semana de 7 dias	32	66	100	134	168
Semana de 8 dias	36	75	114	153	192
Mês de 28 dias	132	267	402	537	672
Mês de 29 dias	136	276	416	556	696
Mês de 30 dias	144	288	432	576	720
Mês de 31 dias	148	297	446	595	744
Ano normal	1752	3504	5256	7008	8760
Ano bissexto	1756	3513	5270	7027	8784

As variáveis L_1 até L_5 constituem as larguras acumuladas de cada patamar do diagrama desde a origem até à intersecção desse patamar com o eixo que representa as horas, como mostra a Figura 4.5. As alturas iniciais dos patamares são definidas como o valor médio entre os pontos horários que esse patamar pretende aproximar. Durante a fase de pesquisa o algoritmo evolui somando ou subtraindo uma hora a um patamar e calculando o erro associado. As alturas são sempre definidas como o valor médio dos valores horários e vão sendo alteradas, devido ao ajuste das larguras, de tal forma que o número de valores associados a cada patamar vai sendo alterado.

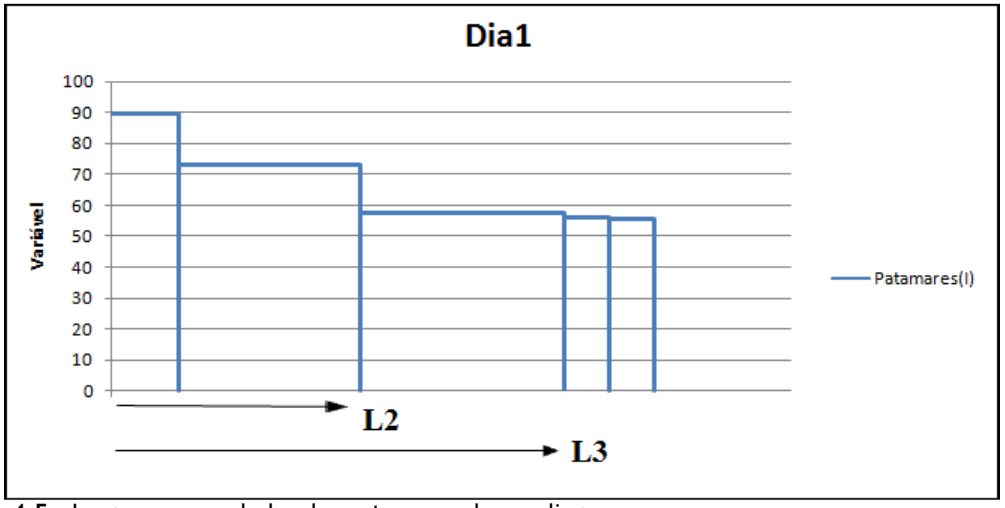

Figura 4.5 - Larguras acumuladas dos patamares de um diagrama.

4.2.4 Cálculo do erro

O cálculo do erro é a parte mais importante do problema, porque é através desse cálculo que são avaliadas soluções em análise.

São várias as abordagens que poderiam ser feitas para calcular o desfasamento que existe entre os pontos horários que a aplicação recebe e os patamares do diagrama que cria. A análise efectuada neste trabalho é descrita de imediato.

O erro considerado, que se pretende minimizar, consiste na área que se estabelece entre a curva que une os valores horários e o patamar mais próximo. Como mostra a Figura 4.6, o erro está associado a uma sucessão de somas de áreas de trapézios.

Seja b_n o valor obtido pela expressão (4.1):

$$b_n = P_n - Patamar_i \tag{4.1}$$

Nesta expressão P_n é o valor horário introduzido na ferramenta. $Patamar_i$ é a altura do patamar *i* que, como já foi referido, é obtido através da média dos pontos que estão mais próximos desse patamar. A expressão (4.2) resume esta explicação, onde *m* é o número total de pontos associados ao patamar *i*.

$$Patamar_i = \frac{\sum_{n=1}^{m} P_n}{m} \tag{4.2}$$

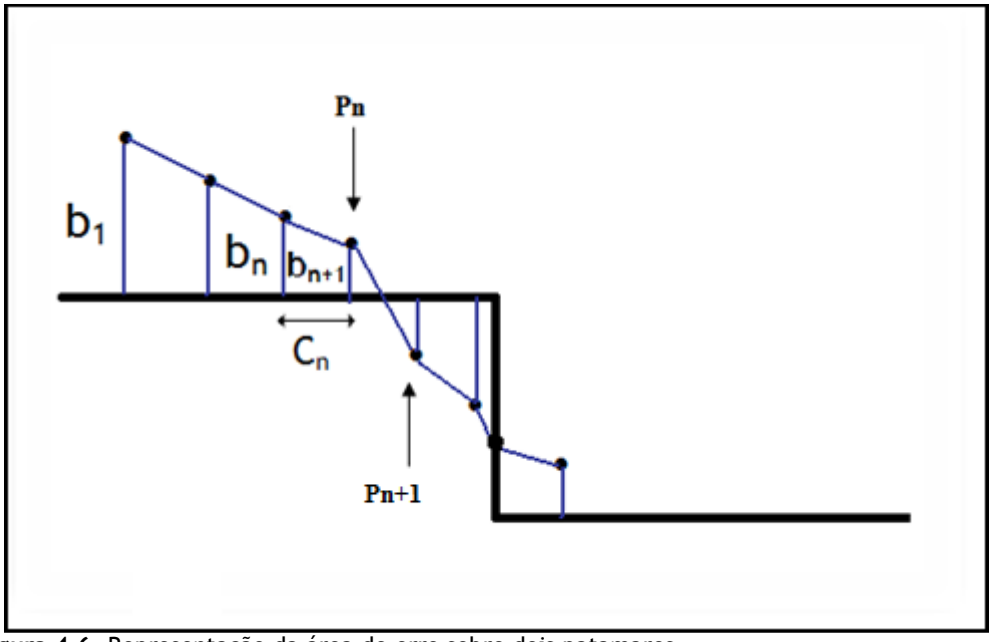

Figura 4.6- Representação da área de erro sobre dois patamares

O cálculo do erro associado a um patamar corresponde à soma das áreas dos trapézios representados e é dividido em duas situações distintas.

Se b_n tiver o mesmo sinal que b_{n+1}:

$$A_i = \sum_{n=1}^{m} \frac{\left(|b_n| + |b_{n+1}|\right).c_n}{2} \tag{4.3}$$

Se b_n não tiver o mesmo sinal que b_{n+1}:

$$A_i = \frac{|b_n|.c_n}{2} + \frac{|b_{n+1}|.(1-c_n)}{2} \tag{4.4}$$

Nestas expressões A_i é a área associada ao patamar *i*. Para este problema, na expressão (4.3) o valor de c_n será sempre 1, uma vez que traduz a distância entre pontos horários. Em qualquer fase de pesquisa do algoritmo este valor manter-se-á sempre em um, excepto quando ocorrer a situação em que entre pontos sucessivos existe a intersecção com o patamar, como é ilustrado na Figura 4.6. A expressão (4.4) indica como é calculada a área nestas situações. O valor de c_n neste caso particular é calculado de acordo com a expressão (4.5).

$$c_n = \frac{-b_n}{b_{n+1} - b_n} \tag{4.5}$$

Estamos agora em condições de apresentar a expressão, que mediante uma solução apresente o valor da sua função de avaliação, ou seja o erro total associado a um diagrama de patamares.

$$Erro = \sum_{i=1}^{5} A_i \tag{4.6}$$

4.2.5 Fluxograma do algoritmo

As Figuras 4.7 e 4.8 mostram o fluxograma do algoritmo utilizado para que a ferramenta alcançasse os objectivos propostos. Como já foi referido, foi usada a técnica *Simulated Annealing*. Seguir-se-á uma breve explicação sobre cada uma das etapas do algoritmo e sobre considerações adoptadas nomeadamente a nível de parâmetros a introduzir no modelo.

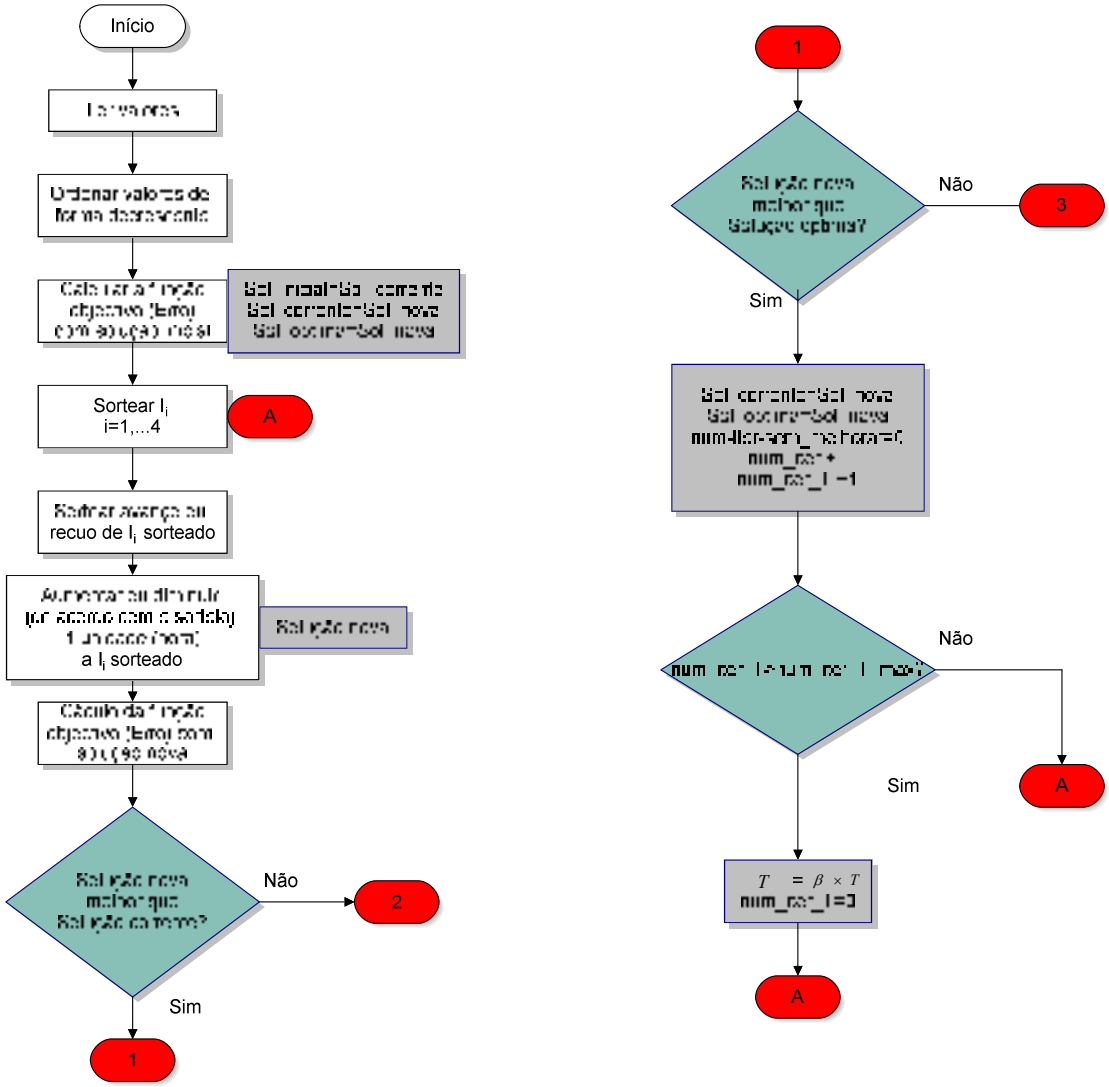

Figura 4.7 - Fluxograma da resolução do problema de optimização em diagramas de patamares (1ªparte)

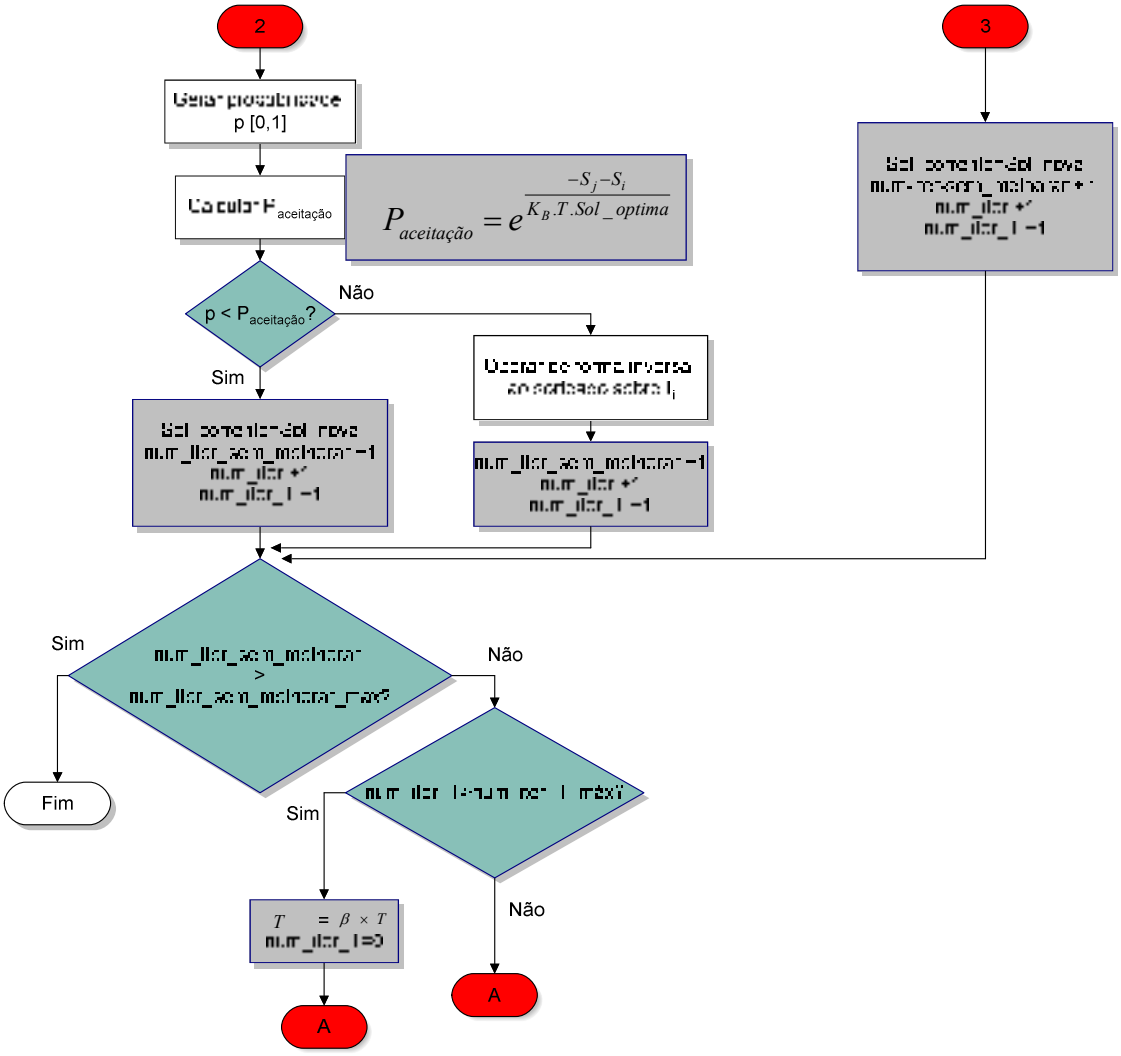

Figura 4.8 - Fluxograma da resolução do problema de optimização de diagramas de patamares (2ªparte)

O algoritmo começa por ler as séries de valores a analisar e ordená-las de forma decrescente, de forma a estarem prontas para formar um diagrama de patamares decrescente. É avaliada a solução inicial através do cálculo do erro associado. Esta solução é guardada como solução corrente, solução nova e solução óptima.

O próximo passo consiste em procurar na vizinhança da solução anterior uma solução melhor. Para isso são definidos 4 pontos, os pontos de união entre patamares I_i. A Figura 4.9 ilustra em pormenor estes pontos, aos quais será alterada a posição, diminuindo ou aumentando um deles de uma unidade (hora). Um destes pontos é então sorteado com o objectivo de ser alterada a sua posição. Alterando a posição de um destes pontos, é modificada a largura de dois patamares adjacentes, e desta forma, é identificada uma nova solução na vizinhança da solução corrente. Depois é sorteado se essa modificação consiste no avanço ou no recuo da posição do ponto seleccionado. Estes sorteios são realizados através da geração de um número aleatório entre 0 e 1 considerando uma distribuição uniforme.

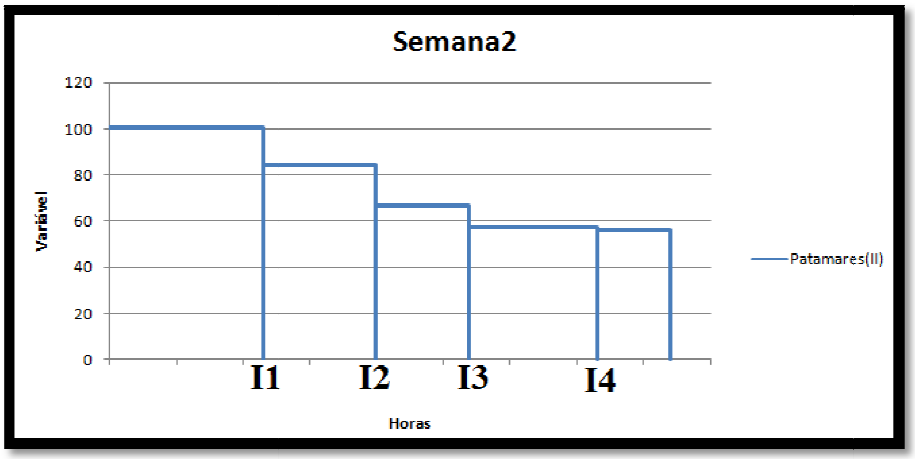

Figura 4.9 - Ilustração dos pontos de união entre patamares, I_i.

Para esta nova solução é calculada a função de avaliação e o seu valor é comparado com o valor que está guardado na variável *Sol_corrente*. Se for melhor, o valor de *Sol_nova* é guardado na variável *Sol_corrente* e se for melhor que a solução óptima do momento é guardada também em *Sol_optima*. No entanto, se a nova solução não for melhor que a solução corrente, pode ainda ser escolhida com uma determinada probabilidade. Essa probabilidade é chamada de probabilidade de aceitação e é calculada segundo a expressão seguinte:

$$P_{aceitação} = e^{\frac{S_j - S_i}{K_B.T.Sol\_optima}} \tag{4.7}$$

Nesta expressão, S_j é a solução nova, S_i é a solução corrente, K_B e T são parâmetros do problema e *Sol_optima* é a variável que guarda o valor da solução óptima do momento. A teoria do *Simulated Annealing* não refere a introdução do valor da solução óptima no cálculo desta probabilidade. No entanto, depois de realizar vários testes, verificou-se que era necessário utilizar uma forma de estandardização para que o algoritmo acompanhasse de forma efectiva qualquer tipo de valor que fosse introduzido no modelo. Desta forma, deixa também de ser necessária a mudança frequente de parâmetros, como a temperatura T e constante de *Boltzman* K_B, que assim estão sempre ajustados ao problema.

O parâmetro temperatura T é de elevada importância para a obtenção de bons resultados. É necessário definir um valor inicial T_0, um coeficiente de arrefecimento β entre 0 e 1 e um número de iterações máximo, *num_iter_T_max*, durante o qual é mantida a mesma temperatura. Quando é atingido esse número, a temperatura é reduzida proporcionalmente ao valor de β, como mostra a expressão 4.8.

$$T = \beta.T \tag{4.8}$$

Posteriormente, o contador de iterações com a mesma temperatura, *num_iter_T*, é colocado em zero e volta-se a efectuar o mesmo ciclo. Este processo é feito para que no início da pesquisa o algoritmo avance directamente para soluções piores mas que poderão estar mais perto do óptimo global ou óptimos locais. À medida que esta pesquisa vai sendo refinada, é reduzida a temperatura o que implica que a probabilidade de aceitação de soluções piores vá diminuindo. Desta forma, são evitados os mínimos locais de fraca qualidade e é garantido que o algoritmo encontra um mínimo local aceitável ou mesmo o óptimo global.

Quando a solução não é aceite segundo o critério probabilístico referido, o algoritmo volta ao ponto de onde partiu imediatamente antes de essa situação ter ocorrido. Ou seja, como a solução vizinha da solução corrente não foi aceite, a pesquisa incidirá agora sobre outra solução vizinha que seja melhor, ou que possa ser aceite segundo a probabilidade de aceitação.

Existe ainda um outro contador de iterações que realiza a contagem apenas das iterações em que solução óptima não é melhorada pelo menos de um valor absoluto de 0,001, *num_iter_sem_melhorar*, e que é colocado a zero sempre que é encontrada uma solução melhor que a óptima do momento. Essa contagem serve como critério de paragem. É estabelecido um número máximo de iterações deste tipo, *num_iter_sem_melhorar_max*, e quando é atingido esse valor o algoritmo pára e disponibiliza a solução óptima do momento.

4.2.6 Ajustes e outras considerações

A EDP Produção indica que a ferramenta desenvolvida deveria ser capaz de analisar vários períodos de tempo em simultâneo. Em especial, era interessante que fosse capaz com vários períodos de tempo do mesmo tamanho minimizar o erro total dos períodos. Nesta minimização era pretendido que fosse cumprida a restrição que indica que as larguras do patamar *i* devem ser iguais em todos os períodos. Esta exigência está relacionada com o modelo VALORAGUA e da forma como esse modelo analisa este tipo de situações. Assim, o programa desenvolvido está preparado para aceitar séries de 5 períodos de igual dimensão em simultâneo, optimizando o erro global, ou seja a soma dos erros associados a cada série.

Numa análise horária para horizontes de tempo por vezes longos, como é esta, há necessidade de certos ajustes. Esses ajustes estão relacionados por exemplo com as mudanças de hora no Outono e na Primavera. Outro ajuste que tem de ser feito numa análise semanal com horizonte anual, é o facto de o número de semanas num ano não ser inteiro. Num ano normal, existem 52 semanas e um dia e num ano bissexto existem 52 semanas e dois

dias. A EDP Produção trata esta questão considerando semanas de 8 dias, que compensam dessa forma o facto de o número de semanas dever ser inteiro. Nesta ferramenta foi introduzida uma opção que permite ao utilizador escolher o tamanho da semana em análise e ajustar os cálculos de acordo com essa opção.

Capítulo 5

Simulações e análise de resultados

Neste capítulo serão analisados resultados de simulações realizadas utilizando a ferramenta criada. Os valores introduzidos foram fornecidos pela EDP Produção e permitem que desta forma sejam demonstradas as diferentes potencialidades da aplicação desenvolvida.

5.1 Primeiro caso de estudo

No primeiro caso de estudo foram utilizados valores da carga em Portugal referente aos primeiros sete dias de Janeiro de 2008. Foram realizadas duas simulações, a primeira com a solução inicial por defeito e a segunda com a solução inicial igual à solução óptima do primeiro caso. Com esta experiência pretende-se comprovar a eficiência do algoritmo.

5.1.1 Simulação A

Partindo da solução inicial para uma semana de valores, tal como indicado no capítulo anterior, o algoritmo foi alterando a largura dos patamares e consequentemente a sua altura ao longo do processo iterativo.

Nesta primeira simulação foi atingida a convergência em 582 iterações. Os resultados obtidos são ilustrados a seguir.

Os parâmetros utilizados neste caso de estudo estão indicados na Tabela 5.1. De notar que apesar da técnica utilizada para estandardizar a probabilidade de aceitação referida no Capítulo 4, os valores dos parâmetros foram ainda assim alterados para diferentes testes. Isto porque se pretendia realizar uma análise mais profunda ao ajuste de parâmetros de forma a identificar os mais adequados para cada problema. Em todo o caso, verificou-se nos vários

estudos realizados que a estandardização feita e a uniformização dos parâmetros conduzem a boas soluções.

Tabela 5.1 - Parâmetros utilizados para o primeiro caso de estudo

Parâmetros	
T_i	1
K_B	0,009
β	0,95

O valor da função de avaliação da solução óptima, ou seja aquela que traduz o estado de energia mínima, foi de 33989,9101. Para entender este valor é ilustrada na Figura 5.1 a evolução do valor da função de avaliação da solução óptima à medida que as iterações iam sendo realizadas.

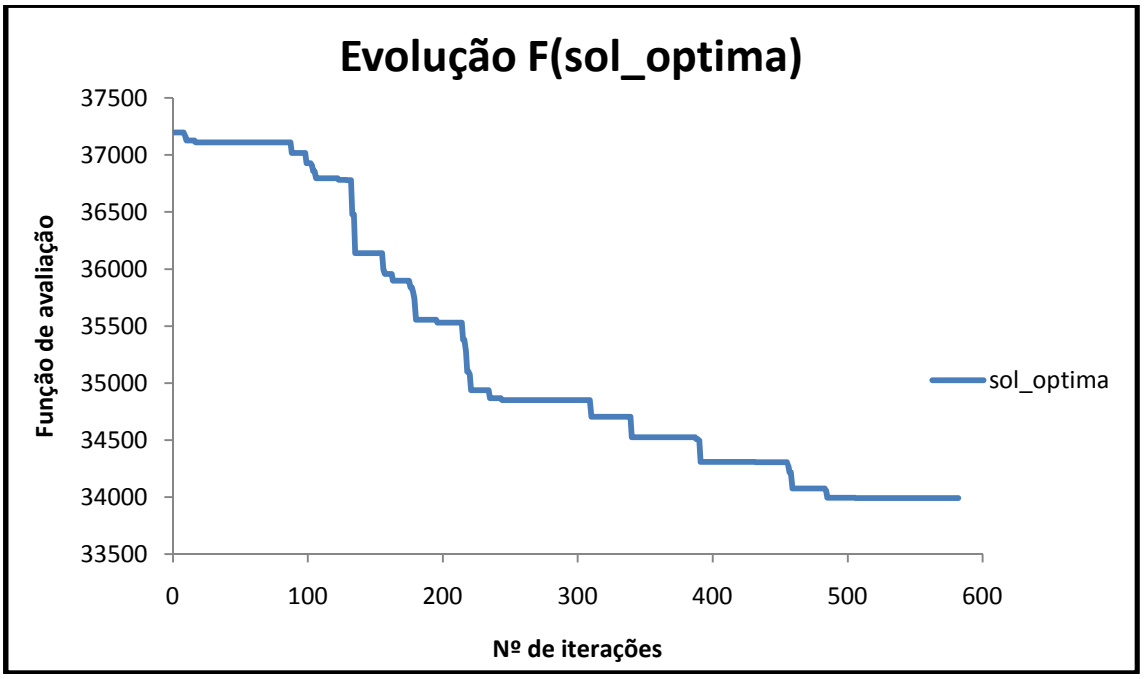

Figura 5.1- Evolução da função de avaliação da solução óptima

Como se pode verificar é garantida uma melhoria de mais de 3000 unidades do erro quando comparada a solução óptima obtida com a solução inicial. A solução que corresponde a este valor de erro é indicada na Tabela 5.2.

Tabela 5.2 - Solução final obtida para o primeiro caso de estudo - Simulação A.

i	L_i (horas)	PAT_i (GW.h)
1	21	8101,709048
2	60	7491,466603
3	95	6493,099714
4	138	5324,991802
5	168	4509,025

Para além de imprimir a solução óptima, a ferramenta cria também o diagrama correspondente à mesma solução. O diagrama correspondente a esta solução é apresentado na Figura 5.2.

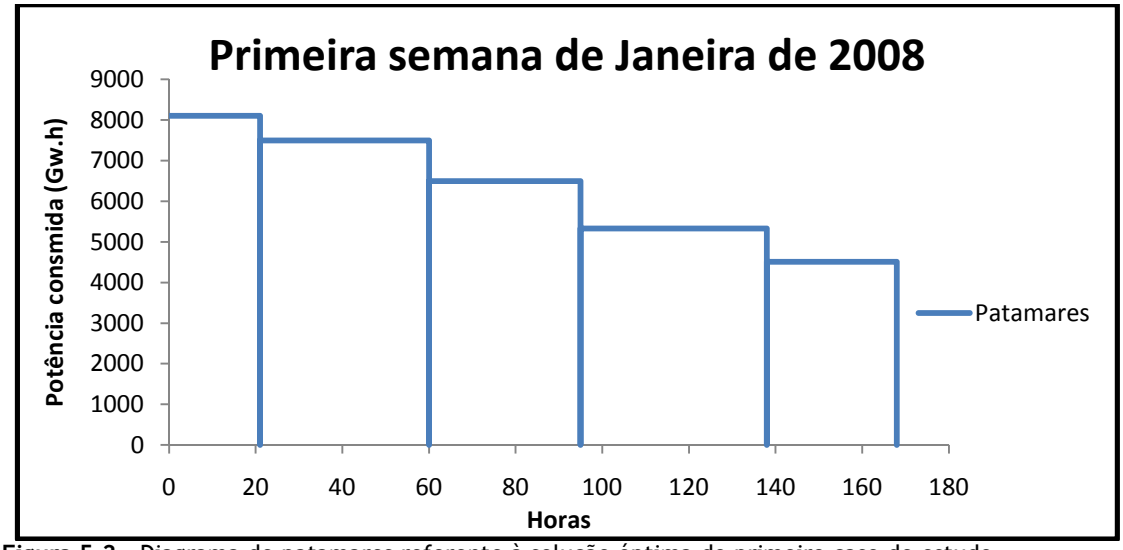

Figura 5.2 - Diagrama de patamares referente à solução óptima do primeiro caso de estudo.

No início do processo de convergência há maior probabilidade de serem aceites piores soluções com objectivo de desviar a pesquisa de óptimos locais para áreas do espaço de soluções que possam proporcionar encontrar uma solução boa. Esse procedimento torna-se mais evidente quando comparadas as evoluções das funções de avaliação das soluções corrente, nova e óptima. A Figura 5.3 ilustra a comparação entre a evolução da função de avaliação da solução nova e da solução corrente.

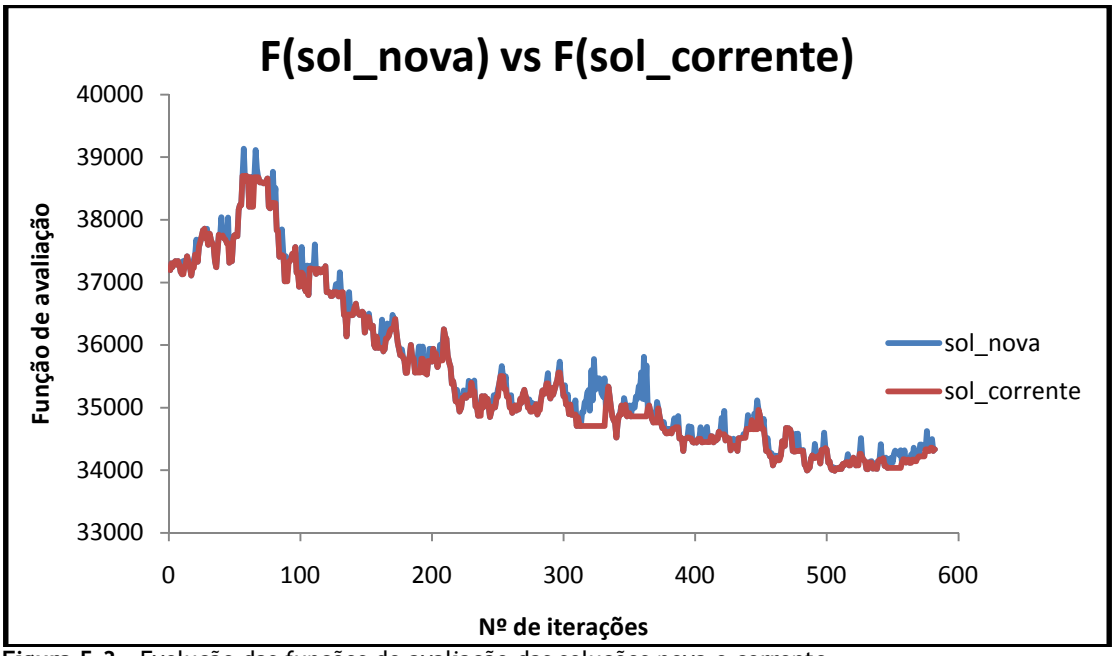

Figura 5.3 - Evolução das funções de avaliação das soluções nova e corrente.

A solução nova é sempre a solução que tem maior valor de erro, uma vez que se for encontrada uma solução melhor que a solução corrente, essa passa a ser a solução corrente. Nota-se que ao fim de algumas iterações a solução corrente já não acompanha tantas vezes a solução nova, isto porque a probabilidade de aceitação de soluções piores é cada vez menor. É interessante também ilustrar a comparação entre a função de avaliação da solução corrente e da solução óptima para complementar a análise. A Figura 5.4 ilustra essa comparação.

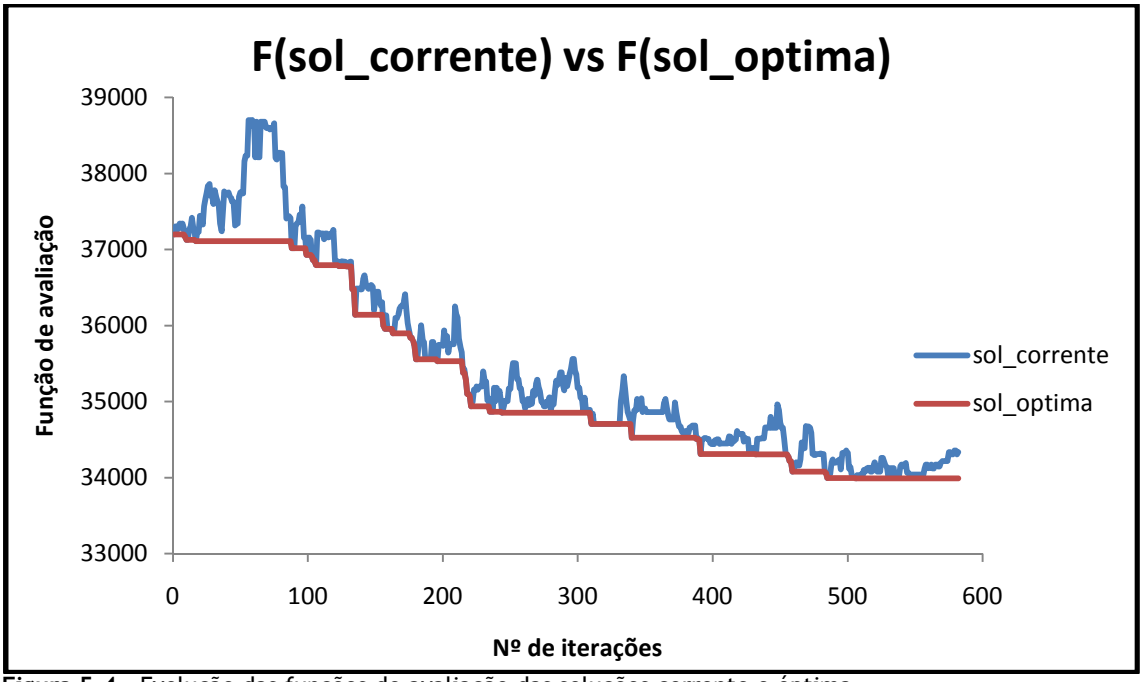

Figura 5.4 - Evolução das funções de avaliação das soluções corrente e óptima.

Nesta figura torna-se mais visível a diminuição da aceitação de soluções piores à medida que a pesquisa vai avançando. Verifica-se que nas 100 primeiras iterações, a pesquisa se

direccionou para soluções muito más, mas isso parece trazer francas melhorias uma vez que, a partir desse ponto, a solução óptima é constantemente melhorada.

Um parâmetro muito importante nesta análise é a Temperatura. Já foram indicados o valor inicial e o parâmetro de arrefecimento. Mas nesta análise é também relevante o número de iterações ao fim do qual a temperatura é reduzida com um factor β. Esse número foi estabelecido em 25 iterações. A Figura 5.5 ilustra esta redução ao longo do processo de pesquisa.

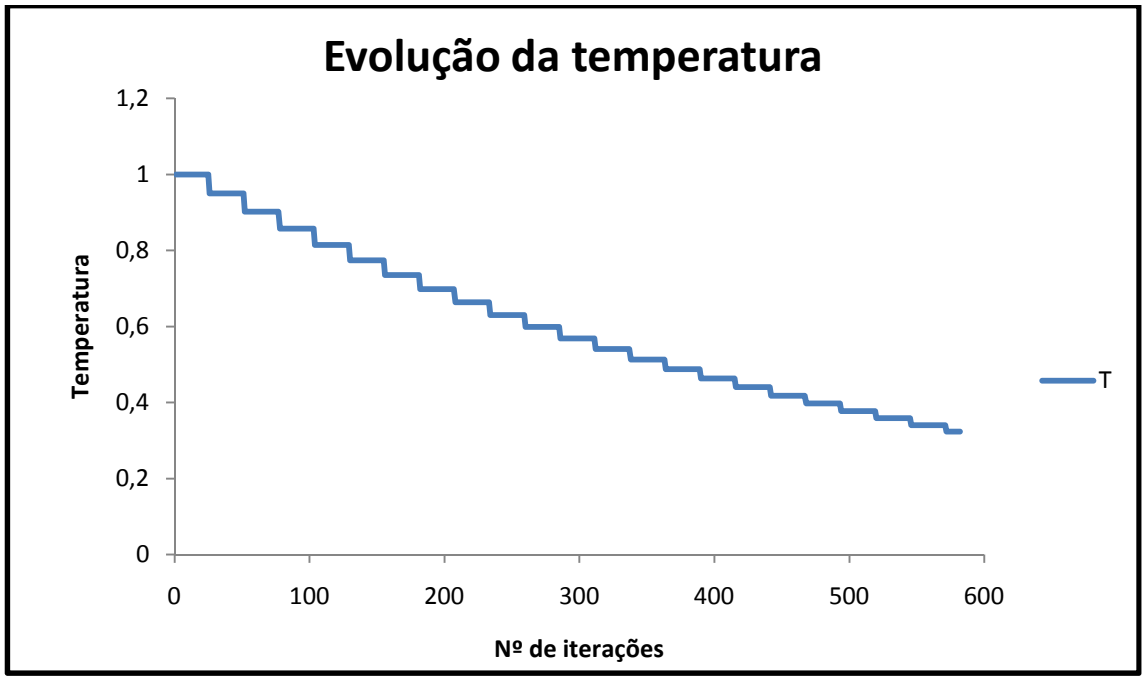

Figura 5.5 - Evolução do parâmetro Temperatura ao longo da pesquisa.

O número de iterações máximo, ao fim do qual o valor da função objectivo permanece sem melhorar pelo menos um número superior a 0.001, foi estabelecido em 75. Este valor é usualmente 3 vezes superior ao número de iterações ao fim do qual se reduz a temperatura.

As larguras dos patamares constituem a variável de controlo deste método. Elas vão sendo alteradas à medida que o processo se desenrola. As alturas dos patamares vão sendo alteradas por consequência, uma vez que consistem na média dos valores horários de cada patamar. As Figuras 5.6 e 5.7 mostram a evolução destas duas componentes durante o processo de pesquisa.

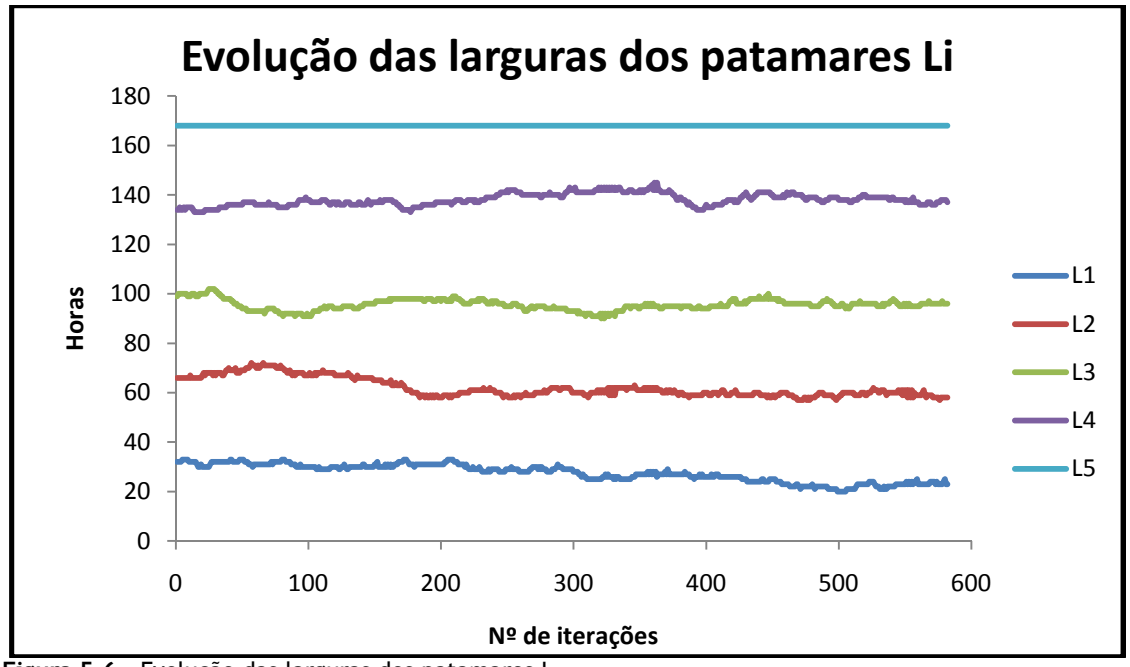

Figura 5.6 - Evolução das larguras dos patamares L_i.

O valor da variável L_5 nunca é alterado uma vez que representa o número de horas total do período de tempo a analisar. São apenas alterados as larguras dos patamares em que há intersecção com patamares adjacentes.

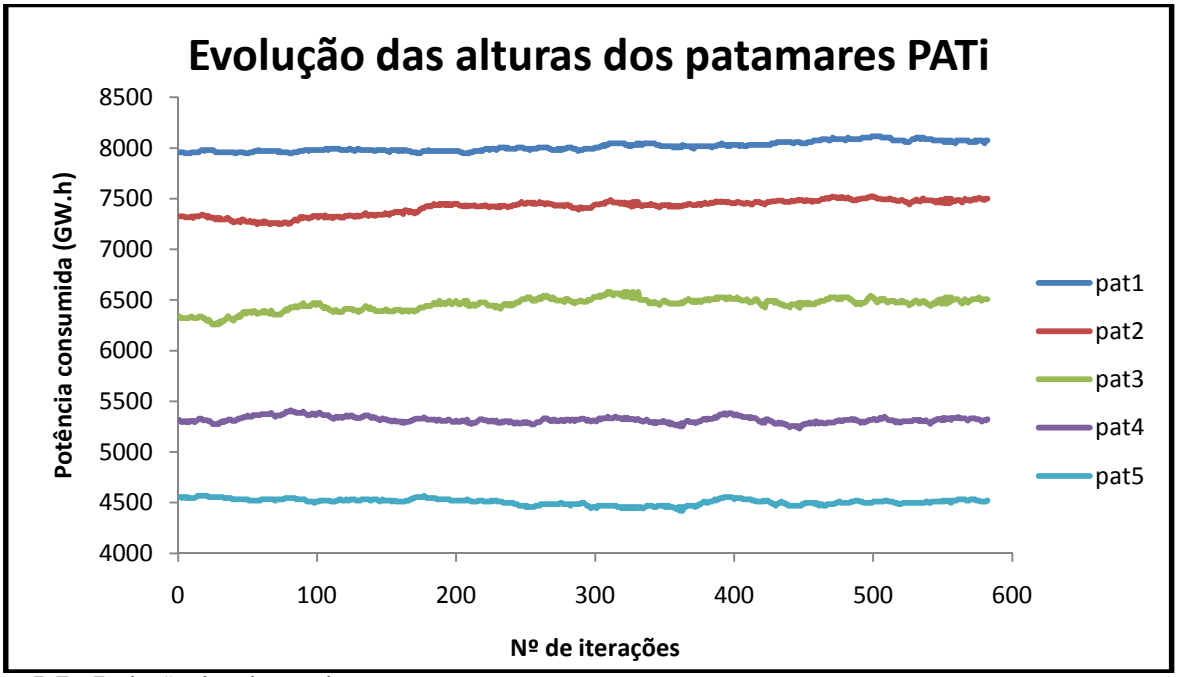

Figura 5.7 - Evolução das alturas dos patamares.

5.1.2 Simulação B

Esta simulação é semelhante à Simulação A. A diferença reside na solução inicial que desta vez é igual à solução final obtida na Simulação A. O interesse desta simulação consiste em verificar se o algoritmo é capaz de encontrar uma solução ainda melhor que a já obtida e assim comprovar a robustez desta técnica e em especial da ferramenta desenvolvida. O gráfico de evolução das funções de avaliação das soluções nova, corrente e óptima é ilustrado na Figura 5.8.

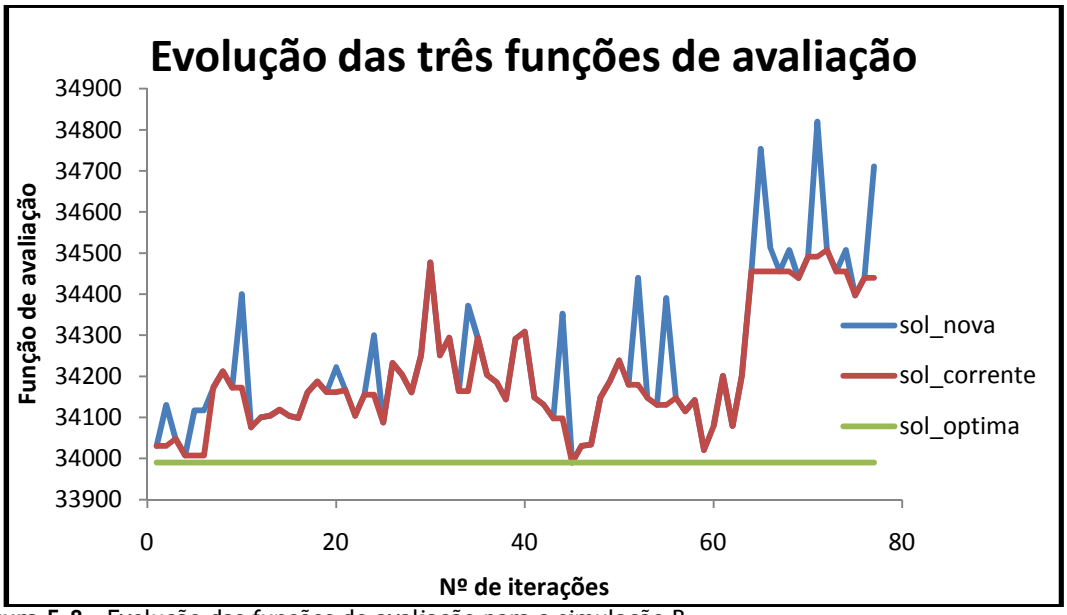

Figura 5.8 - Evolução das funções de avaliação para a simulação B

Como se pode conferir, a solução óptima não sofre modificações e, por isso, o processo converge rapidamente. Isto certifica a robustez e eficiência do método. Deve ser referido, no entanto, que poderiam ocorrer situações em que a solução óptima fosse ainda melhorada, sem retirar qualquer mérito ao método utilizado e à qualidade dos parâmetros escolhidos. Por outro lado, se a solução inicial da Simulação A estivesse já muito próxima da solução óptima global, a pesquisa poderia ser levada para longe desta e ficar aí retida. Este possível afastamento deve-se ao facto da probabilidade de aceitar soluções piores ser alta no início do processo de pesquisa e poderia fazer com a pesquisa se perdesse e já não tivesse retorno a uma solução de qualidade.

5.2 Segundo caso de estudo

O segundo caso de estudo utiliza os valores de carga em Portugal para quatro dias distintos, 1 de Fevereiro, 1 de Março, 1 de Abril e 1 de Maio de 2008. Esta simulação tem a particularidade de mostrar as capacidades da ferramenta quando se pretende optimizar as

larguras dos patamares de vários diagramas em simultâneo. Esta optimização é realizada baseada na restrição de que a largura do patamar *i* deve ser igual em todos os diagramas. Esta restrição é necessária para que esta ferramenta interaja com o modelo VALORAGUA. No entanto, será descrita uma outra simulação, Simulação B, em que são consideradas 4 optimizações distintas e independentes para comparação e análise.

5.2.1 Simulação A

Nesta simulação não houve mudança de nenhum dos parâmetros adoptados anteriormente. O número de iterações ao fim do qual se reduz a temperatura e o número ao fim do qual a convergência é atingida, também se mantiveram os mesmos. O erro global da solução encontrada foi de 11511,96. A Tabela 5.3 contém os valores obtidos no fim da pesquisa. De uma forma compacta, a Figura 5.9 ilustra os diagramas que representam a solução final obtida.

Tabela 5.3- Solução final obtida para o segundo caso de estudo - Simulação A

i	L_i	PAT_i (GW.h)	PAT_i II (GW.h)	PAT_i III (GW.h)	PATi IV (GW.h)
1	5	7814,38125	7066,456	7060,161	6641,38
2	10	7414,063125	6703,21	6669,18	6304,163125
3	17	6972,635	6327,676	6224,804	5809,735
4	19	5595,6825	5171,717	4842,001	4806,725
5	24	4943,935625	4661,974	4237,657	4404,22125

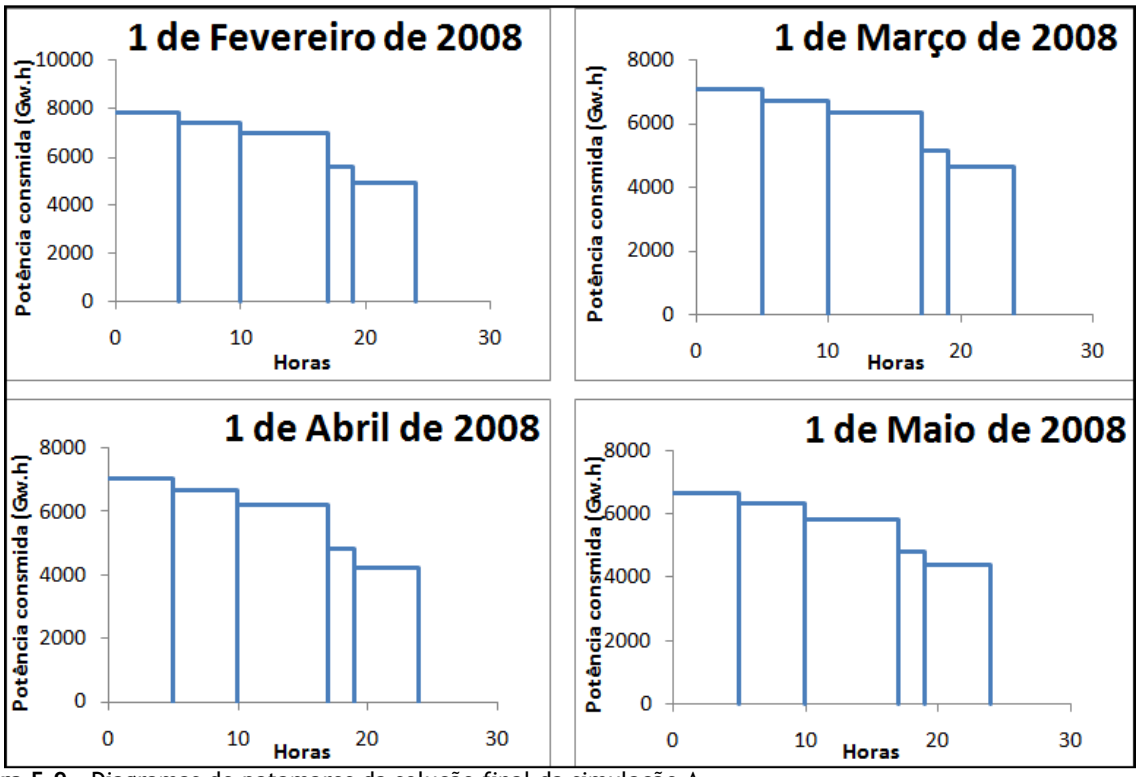

Figura 5.9 - Diagramas de patamares da solução final da simulação A

O processo convergiu em 191 iterações e a evolução da função de avaliação da solução óptima ao longo da pesquisa está representada na Figura 5.10.

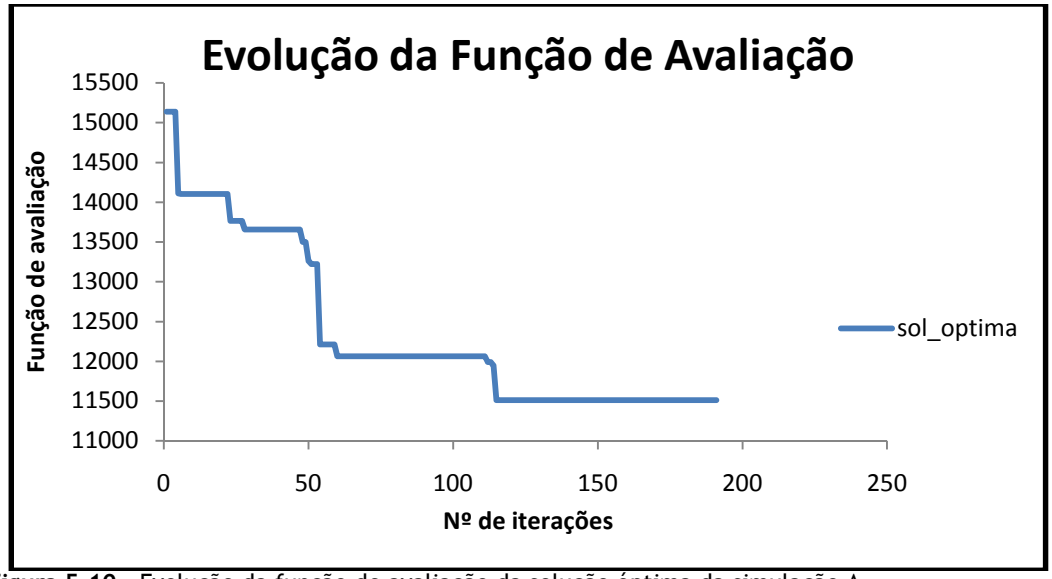

Figura 5.10 - Evolução da função de avaliação da solução óptima da simulação A.

A solução inicial tinha um erro de 15138.52 e foi substancialmente melhorada. Entre a iteração 60 e 115 a função de avaliação estabiliza e mantém o mesmo valor. É mostrado assim que o valor do número máximo de iterações sem ocorrer melhoria da função de avaliação estabelecido em 75 não é um valor exagerado, porque depois desse período de estabilização a solução é ainda melhorada.

5.2.2 Simulação B

Esta simulação consiste na verdade em várias simulações. Foram simuladas optimizações distintas para os 4 dias em análise de forma independente. Desta forma será analisado se há degradação da solução optimizando períodos de tempo em simultâneo em vez de se realizarem optimizações independentes. Mais uma vez os parâmetros do problema continuam a ser os mesmos, assim como o número de iterações máximo com a mesma temperatura e sem ocorrerem melhorarias.

A Tabela 5.4 apresenta o valor da função objectivo da melhor solução encontrada e o número de iterações para cada período de tempo.

Tabela 5.4- Valores individuais de função de avaliação de soluções finais obtidas para o segundo caso de estudo – Simulação B.

Dia	F (solução óptima)	Nº de iterações
1 de Fevereiro	3194,888	259
1 de Março	2669,436	127
1 de Abril	2859,256	96
1 de Maio	2757,246	121
Total	**11480,83**	

O erro total é, como esperado, menor que o erro global obtido na Simulação A. Sem restrições ligadas ao valor das larguras dos patamares é normal que a soma das optimizações individuais traduza este resultado. Este resultado não é no entanto significativamente melhor que o anterior. Há que ter em conta que existem várias condicionantes ligadas ao processo de convergência, como parâmetros do problema e uma certa aleatoriedade, que podem condicionar um resultado mais rigoroso.

A Figura 5.11 ilustra a evolução do erro inerente às soluções finais dos 4 períodos de tempo considerados.

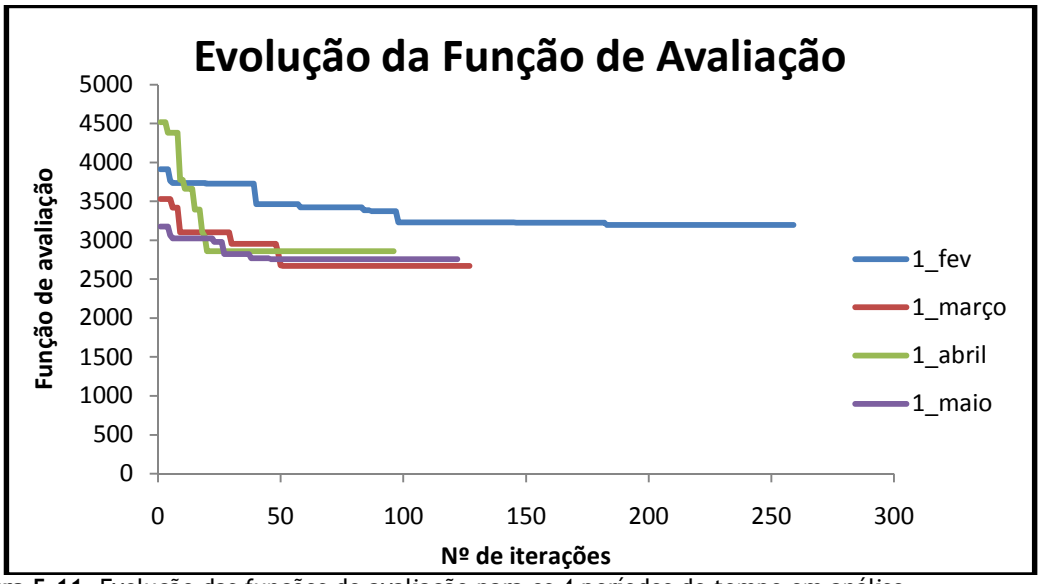

Figura 5.11- Evolução das funções de avaliação para os 4 períodos de tempo em análise.

Todos os processos iterativos acabam por convergir, embora de forma diferente como se pode observar. As soluções iniciais são iguais em todos os períodos, no entanto o valor da função de avaliação dessa solução é diferente em cada período. É assim realçada a importância que a solução inicial desempenha no desenrolar do processo de convergência. Nota-se igualmente que apesar da solução inicial ser eventualmente má, a pesquisa pode encontrar uma solução boa ou mesmo a solução óptima global. Uma situação desse género ocorre com a optimização feita para o dia 1 de Abril, com ilustra a Figura 5.11.

As Tabelas 5.5 e 5.6 apresentam as soluções finais obtidas, nomeadamente as larguras e alturas finais individuais.

Tabela 5.5 - Larguras dos diagramas independentes referentes à solução final obtida para o 2º caso de estudo – Simulação B.

	Larguras L_i			
i	**1 de Fevereiro**	**1 de Março**	**1 de Abril**	**1 de Maio**
1	5	4	4	4
2	15	15	11	11
3	18	17	16	15
4	20	21	19	19
5	24	24	24	24

Tabela 5.6 - Alturas dos diagramas independentes referentes à solução final obtida para o 2º caso de estudo – Simulação B.

	Alturas PAT_i			
i	**1 de Fevereiro**	**1 de Março**	**1 de Abril**	**1 de Maio**
1	7764,75	6925,31968	7060,16062	6641,38
2	7309,84425	6600,135	6703,33071	6343,02714
3	6113,70583	6070,63812	6265,8675	5962,95812
4	5289,995	5067,76416	5079,512	5029,81437
5	4943,93562	4661,97375	4359,93458	4428,75

5.3 Terceiro caso de estudo

Para o terceiro caso de estudo foram usados preços horários de electricidade em Portugal e em Espanha. O horizonte temporal é um ano, o ano de 2008. Serve este exemplo para demonstrar as potencialidades da ferramenta criada quando sujeita a uma optimização onde são incluídos muitos valores e para permitir fazer uma análise sobre a diferença de preços praticadas nos dois países. Todos os parâmetros e número de iterações máximos foram mantidos em relação aos exemplos anteriores.

A Figura 5.12 ilustra a solução final encontrada para as duas simulações executadas em forma de diagrama de patamares. Deve-se assinalar que foram realizadas duas optimizações separadas, uma para os preços verificados em Portugal e outra para Espanha, de que resultaram larguras e alturas dos patamares diferentes como se pode observar na Figura 5.12.

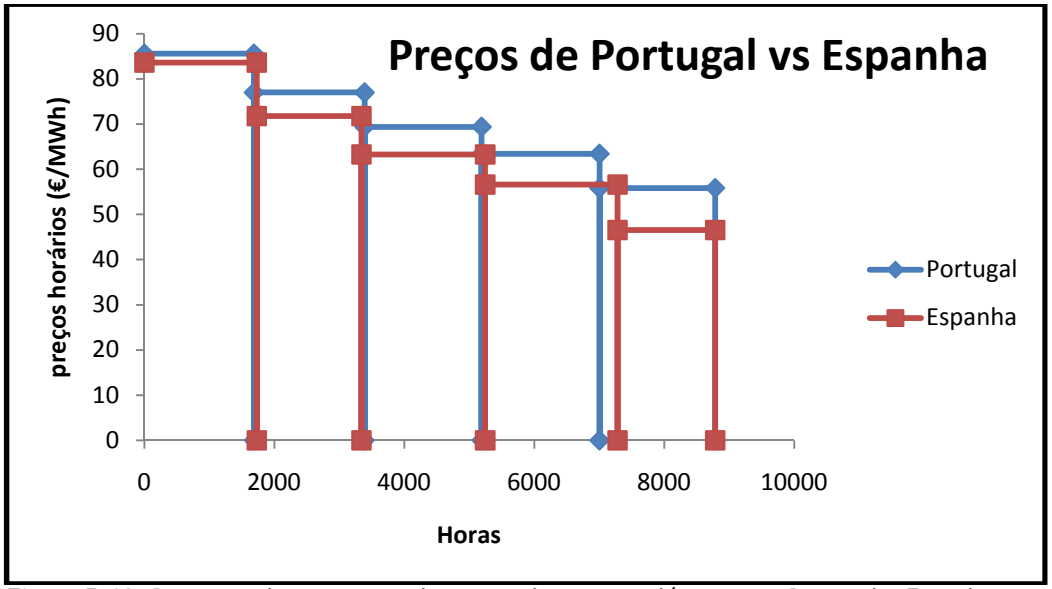

Figura 5.12- Diagrama de patamares de preços de energia eléctrica em Portugal e Espanha.

Os dados relativos ao comportamento do processo de pesquisa são agora indicados na Tabela 5.7.

Tabela 5.7 - Dados relativos à execução do algoritmo no 3º caso de estudo.

Dados introduzidos	Nº de iterações	F (solução óptima)	F (solução inicial)
Preços de Portugal	1838	20019,38786	20281,70774
Preços de Espanha	3162	24244,66	24740,94

Mais uma vez é visível a importância da solução inicial. A mesma solução inicial, indicada no Capítulo 4, tem um valor diferente em cada uma das simulações, o que pode ter contribuído decisivamente para uma diferença substancial entre o valor da solução final em cada um dos casos.

O diagrama da Figura 5.12 mostra que os preços de Portugal foram, regra geral, mais elevados que em Espanha no ano de 2008. Esta diferença de preços está certamente relacionada com a diferença de produção nuclear e produção em regime especial existente nos dois países. A Tabela 5.8 resume as características da solução final obtida para cada um dos casos e clarifica ainda mais esta diferença.

Tabela 5.8 - Soluções finais das simulações com preços de Portugal e Espanha correspondentes ao 3º caso de estudo.

i	Portugal		Espanha	
	Li (horas)	PATi (€/MWh)	Li (horas)	PATi (€/MW.h)
1	1686	85,57134	1730	83,61171
2	3388	76,98063	3339	71,74982
3	5189	69,37279	5243	63,25829
4	7004	63,39684	7282	56,61232
5	8784	55,82699	8784	46,55164

5.4 Quarto caso de estudo

Este último caso de estudo envolve valores de carga e valores de preços de electricidade. Foram feitas simulações com os dois tipos de dados para posteriormente comparar as soluções finais de forma a tentar encontrar possíveis desfasamentos horários entre o diagrama de cargas e o diagrama de preços. Para isso, na ferramenta criada foi desactivado o módulo que ordenava de forma decrescente os valores introduzidos.

Foram feitas duas simulações para cada tipo de dados, uma com valores para uma semana, outra para valores para um dia.

5.4.1 Simulação A

Para os preços horários de electricidade e carga em Portugal na semana de 20 a 26 de Abril de 2008, foram feitas optimizações do diagrama de patamares correspondente. Como já se disse, não houve desta vez uma ordenação decrescente de valores, para realçar onde se situa cada gama de valores.

As soluções finais obtidas para os preços e para as cargas são ilustrados na Figuras 5.13 e 5.14 respectivamente.

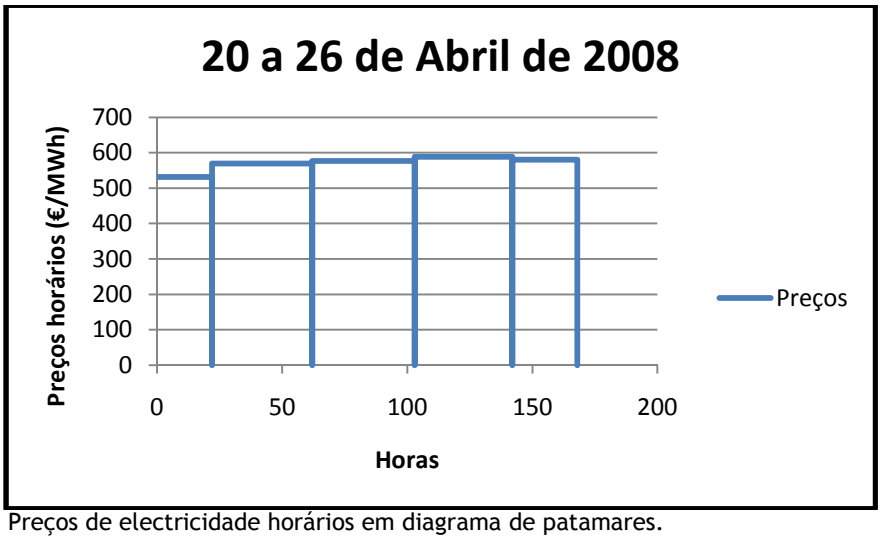

Figura 5.13 - Preços de electricidade horários em diagrama de patamares.

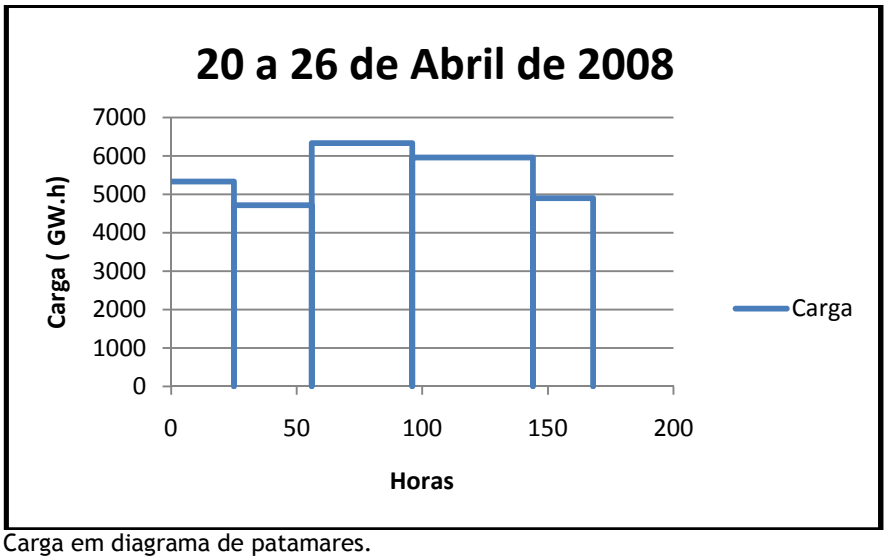

Figura 5.14 - Carga em diagrama de patamares.

Pode-se observar que não há uma ligação directamente proporcional entre a carga e os preços nos mesmos dias. Tal deve-se ao facto de os preços de electricidade não serem calculados unicamente a partir do valor da carga, mas também dependerem de diversos outros factores que os influenciam.

5.4.2 Simulação B

Depois de observados os valores para uma semana, desta feita serão observados os valores para um dia, o dia 21 de Abril de 2008 que pertence à semana analisada na Simulação A. As Figuras 5.15 e 5.16 ilustram os diagramas finais obtidos para os preços e para as cargas.

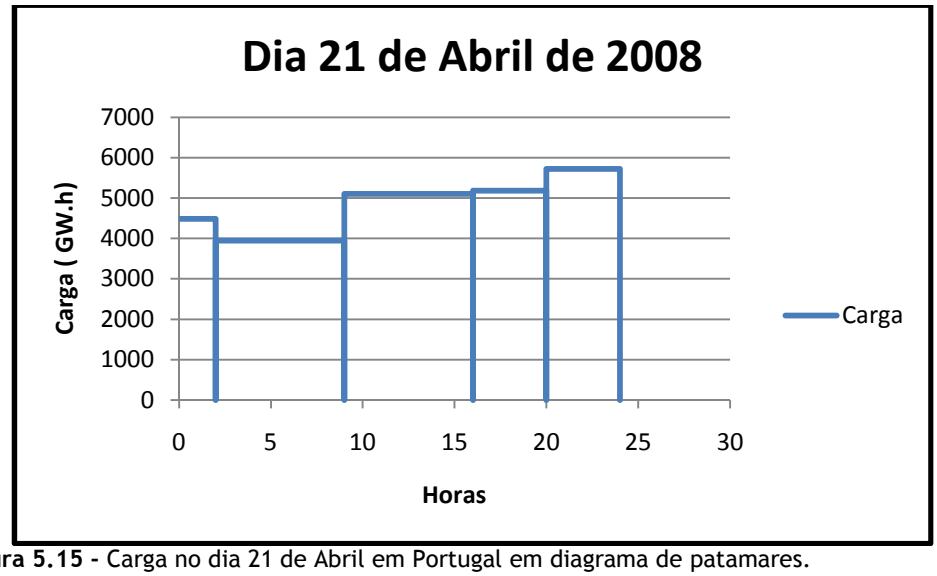

Figura 5.15 - Carga no dia 21 de Abril em Portugal em diagrama de patamares.

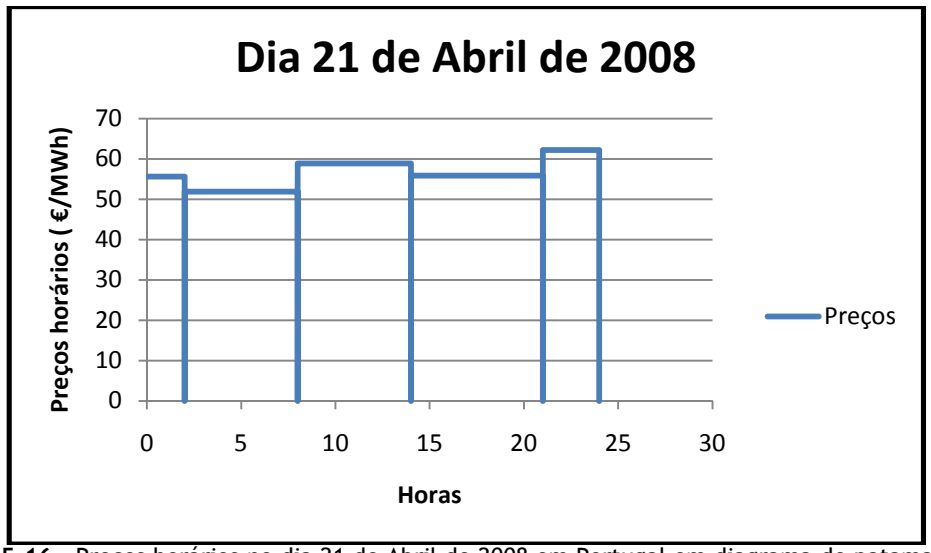

Figura 5.16 - Preços horários no dia 21 de Abril de 2008 em Portugal em diagrama de patamares.

Como se pode observar o diagrama de patamares referente aos preços de electricidade acompanha o das cargas um pouco melhor que no exemplo anterior. No entanto, é visível que há ainda diferenças de andamento substanciais referindo a existência de outros factores a influenciar o preço de electricidade.

5.5 Comentários

Os quatro casos de estudo apresentados visaram ilustrar da melhor forma as potencialidades da ferramenta criada, tanto nos valores que admite como entradas como as possíveis análises que com ela podem ser realizadas.

As simulações são regra geral rápidas, sendo o seu tempo de convergência médio de 2 minutos, numa máquina com processador de 2,00 GHz e de 2GB de memória RAM. É possível que existam parâmetros mais ajustados aos problemas que aqui foram expostos. No entanto, não se pretende que estes sejam alterados de problema para problema, pelo que se optou pela utilização da estandardização referida atrás. Estes parâmetros mostraram que são capazes de conduzir a boas soluções, e foram testados com uma grande gama de valores de diferentes variáveis. A estandardização tem como grande vantagem tornar mais fácil e rápida a utilização da aplicação desenvolvida por qualquer utilizador, pelo que constitui um bom compromisso entre a qualidade da solução inicial e a rapidez e facilidade de utilização da aplicação.

Capítulo 6

Conclusões

Este projecto surgiu no âmbito dos estudos de planeamento a médio e longo prazo do sector eléctrico e das necessidades inerentes a esse planeamento. Neste âmbito, surgiu a necessidade da criação de uma aplicação computacional que fosse flexível a vários tipos de análises e que interagisse com o modelo VALORAGUA e com todas as suas condicionantes. O tema a abordar foi referido pela EDP Produção tendo-se revelado como uma boa oportunidade de interacção com a indústria e com as suas necessidades.

A ferramenta criada permite optimizar diagramas de patamares, ajustando-os a uma série de valores inseridos, tipicamente de energia consumida ou de preços de electricidade. Devido ao carácter altamente combinatório do problema, foi decidido utilizar uma metaheurística. Esta técnica é capaz de escapar a óptimos locais, realizando uma pesquisa robusta do espaço de soluções.

Foi realizada uma ampla pesquisa bibliográfica sobre estes métodos, em especial sobre os que foram explanados neste trabalho. O *Scatter Search* usa pontos de referência que constituem boas soluções obtidas a partir de soluções anteriores. O *Path Relinking* gera novas soluções explorando trajectórias que unem soluções de alta qualidade, gerando um caminho na vizinhança que direcciona o processo iterativo para outras soluções. O *Tabu Search* segue a pesquisa local sempre que esta encontre um óptimo local, sendo permitido movimentos que não melhoram a função de avaliação. Para além disso, o retorno cíclico a soluções pré-visitados é impedido através da utilização de memórias que recordam a história recente da pesquisa, chamadas de listas tabu. As desvantagens desta técnica consistem no aparecimento de atractivos por consequência da exploração da vizinhança e o incómodo que é a propensão à descida muito rápida em direcção ao óptimo local. Os Algoritmos Genéticos são baseados em mecanismos de selecção natural e de operadores genéticos. Estes aplicam uma estratégia de procura paralela e estruturada, mas aleatória, que é direccionada ao reforço da procura

de selecções com boa qualidade. O *Variable Neighborhood Search* é baseado na mudança sistemática de vizinhança durante a pesquisa. Uma grande vantagem desta técnica consiste no facto do óptimo local encontrado no fim da pesquisa dizer respeito a todas as vizinhanças e, por isso, as hipóteses de alcançar um óptimo local são maiores que em outros métodos. O *Guided Local Search* usa determinadas características do problema e associa um custo e uma penalidade a cada uma. O custo é definido através de uma função objectivo e pelos seus coeficientes. As penalidades são incrementadas se a pesquisa alcançar um óptimo local. O *Greedy Randomized Adaptive Search Procedure* é um processo iterativo, em que cada iteração inclui duas fases: a construção de soluções admissíveis e a pesquisa local para melhorar essas soluções. Tem a desvantagem de necessitar de boas soluções iniciais para ter sucesso nos resultados e a vantagem de permitir o uso de filtros que aceleram a pesquisa local. Os algoritmos baseados em Colónias de Formigas foram buscar a sua inspiração ao rasto de feromonas que as formigas reais deixam como forma de comunicação entre elas, para encontrar o melhor caminho entre dois pontos. De referir que o modelo computacional que implementa esta ideia não é muito semelhante ao modelo de colónias naturais de formigas. Isto porque, por exemplo, as formigas deixam o rasto de feromonas durante o seu movimento, e não depois de alcançar o seu destino final.

Finalmente, o Simulated Annealing simula o processo físico de submeter um material a uma temperatura elevada, com consequente arrefecimento, de forma a obter cristais de alta qualidade. Um parâmetro de elevada importância é a Temperatura, porque através dela praticamente se define se o algoritmo aceita mais ou menos soluções de pior qualidade. É através desta manipulação que os óptimos locais são evitados. Esta técnica foi a usada neste trabalho por vários motivos. É uma técnica de fácil implementação e com bons resultados obtidos em vários projectos com diferentes abordagens. A grande dificuldade de adaptar esta técnica ao problema resulta da manipulação dos parâmetros, nomeadamente os valores para a Temperatura inicial, coeficiente de arrefecimento, número de iterações ao fim das quais a temperatura decresce e critério de convergência. No que se refere ao trabalho desenvolvido, a técnica utilizada mostrou bons resultados em tempos muito curtos, especialmente tendo em conta que se trata de um problema de carácter combinatório em que existe uma grande variedade de soluções possíveis.

Em relação ao desenvolvimento da aplicação computacional é importante referir que foi dada grande importância à criação do módulo que calcula o erro. Isto deve-se ao facto de este ser o primeiro passo para que a ferramenta consiga alcançar bons resultados. O erro corresponde ao valor da função de avaliação do *Simulated Annealing* determinando se uma solução é melhor ou pior. Foi decidido que o erro seria calculado somando áreas entre os valores a aproximar e os patamares do diagrama de patamares. Não foi esquecido, no

entanto, que este tinha de ser calculado para que os erros não se anulassem, ou seja quando adequado, foram introduzidos valores absolutos. Quanto à utilização da técnica *Simulated Annealing* tiveram de ser tomadas decisões, nomeadamente em relação aos parâmetros a utilizar. Depois de vários testes efectuados, chegou-se à conclusão que uma forma de estandardizar o problema era o de colocar o valor da função de avaliação da melhor solução obtida até cada iteração no cálculo da probabilidade de aceitação. Isto permitiu que os parâmetros não tivessem de estar constantemente a ser ajustados, à medida que se iam alterando os tipos de valores introduzidos. Os dados fornecidos pela EDP Produção foram bastante úteis, não só para permitirem demonstrar as capacidades do algoritmo implementado, mas também por permitirem uma gama alargada de resultados.

Para melhorar esta ferramenta, como trabalho futuro sugere-se que o algoritmo seja capaz de analisar mais diagramas de patamares em simultâneo, alguns melhoramentos a nível do interface com o utilizador e que seja implementada a mesma ferramenta com uma outra técnica para comparar a robustez do método.

Dito isto, deve-se referir que foram alcançados os objectivos inicialmente propostos e que a aplicação computacional está pronta a ser utilizada em ambiente empresarial na EDP Produção a quem se agradece, de novo, a oportunidade de realizar este trabalho.

Referências

1. Alba, E. and B. Dorronsoro, *Cellular Genetic Algorithms* Operations Research, Computer Sciences Interfaces, ed. R. Sharda, O.S. University, and O. Stillwater, USA. pp. 4-7,2008, Málaga, Spain: Springer.
2. Glover, F. and G.A. Kochenberger, *Handbook of Metaheuristics*. Operations Research & Management Science, ed. F.S. Hillier. pp.1-55 , pp. 145-219, 2003, Dordrecht: Kluwer Academic.
3. Tomassini, M., *A survey of Genetic Algorithm*. Vol. II Anal Reviews of Computational Physics. 1998: World Scientific.
4. de Noronha, T.F., *Uma abordagem sobre estratégias metaheurísticas*, in *Departamento de Informática de Matemática Aplicada*, Universidade Federal do Rio Grande do Norte.
5. Quesado, P., *Coordenação de relés de máximo de intensidade homopolares e homopolares direccionais ultilizando o algoritmo evolucionário EPSO*, in *Engenharia Electrotécnica e de Computadores*. 2008, Faculdade de Engenharia da Universidade do Porto: Porto, pp. 35-37.
6. Holland, J.H., *Adaptation in natural and artificial systems* 1975, University of Michigan Press.
7. Goldberg, D.E., *Genetic Algorithms In Search, Optimization And Machine Learning*.Addison-Wesley Professional; 1st edition 1989.
8. Hopgood, A.A., *Intelligent Systems forEngineers and Scientists*. CRC Press; 2 edition, 2001. ISBN:0-8493-0456-3.
9. Zebulum, et al., *Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*, Evolutionary Electronics. pp.39-44. CRC Press. 2002
10. Miranda, V., *Computação Evolucionária Fenotipica*. Março, 2005.
11. Mladenovic, N., *A variable neighborhood algorithm - a new methaheuristic for combinatorial optimization*. 1995, Montréal p112.
12. Mladenovic, N. and P. Hansen, Variable Neighborhood Search,*Computers operations research*, Vol. 24, pp.1097-1100. 1997.
13. Feo, T.A. and M.G.C. Resende, *A probabilistic heuristic for a computationally difficult set covering problem*. Operations Research Letters, Vol. 8, pp. 67-71.1989.
14. Feo, T.A. and M.G.C. Resende, *Greedy randomized adaptive search procedures*. Journal of Global Optimization, Vol. 6,pp. 109-133.1995.
15. Dorigo, M., G.D. Caro, Vol. 5(2),pp. 137-172. 1999.
16. Dorigo, M. and G.D. Caro, *The Ant Colony Optimization meta-heuristic, New ideas in optimization*, pp.11-32. 1999, London, UK.
17. Dorigo, M. and T. Stützle, *Ant Colony Optimization*.University of Massachusets, London, England, pp.9-47. 2004.

18. Lee, K.Y. and M.A. El-Sharkawi, *Modern Heuristic Optimization Techniques* ed. M.E.E.-. Hawary. 2008: John Wiley & Sons, Inc., Hoboken, New Jersey.pp. 123-144.
19. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, *Optimization by Simulated Annealing,* Science, Vol. 220, No. 4598, pp. 671-680, 1983.
20. Viana, A., J.P.d. Sousa, and M. Matos, *Simulated annealing for the unit commitment problem*, Power Tech Proceedings, 2001 IEEE Porto, Volume 2, 10-13 September 2001, pp:4 pp Vol.2.
21. Simopoulos, D. and S. Kavatza, *Consideration of ramp rate constraints in unit commitment using simulated annealing*, Power Tech, 2005 IEEE Russia,27-30 Jun 2005 pp. 1-7.
22. Braga, A.S.D. and J.T. Saraiva, *Long Term Transmission Expansion Planning - a simulated annealing based multiyear algorithm including long term marginal prices*, in *8th International Conference on Probabilistic Methods Applied to Power Systems*. 12-16 September 2004. pp. 551-556: Iowa State University, Ames, Iowa, IEEE.
23. Hyang, C.-L. and A.S.Masud, *Multiple objective decision making - methods and applications - a state of the art survey, lecture notes in economics and mathematical systems*. 1979, Berlim: springer - verlag.
24. Filho, C.F.F.C., A.T.de Albuquerque, and M.G.F. Costa, *Luminance Optimization in Closed Environments by simulated annealing*, Natural Computation 2008. ICNC '08. Fourth International Conference on Volume 1, 18-20 Oct. 2008 pp. 496-501
25. Filho, J.M., *Electric Industrial Installation*. 7º ed. Rio de Janeiro,: LTC S/A, 2007, 914p.
26. Tavares,M. Natália and Vilela, Sonia, *The importance of hydrolectric resource management in Electricity Market-VALORAGUA model*, Disponível em http://aerna2006.de.iscte.pt/papers/Workshop_Tavares.pdf