Draft of paper In: Revista de Investigação Operacional, 21 (2), Dez (2001) pp 205-231 – In Portuguese

Utilização de *Simulated Annealing* em *Optimização Difusa*

Maria Leonilde Rocha Varela
Universidade do Minho
Escola de Engenharia,
Dept. Produção e Sistemas
Campus de Azurém, 4800-058, Guimarães, Portugal
email: leonilde@dps.uminho.pt

Rita Almeida Ribeiro
Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia,
Dept. Informática
2825-114 - Monte de Caparica, Portugal
email: rr@di.fct.unl.pt

Abstract

The Simulated Annealing algorithm (SA) is an adequate tool for solving fuzzy optimisation problems, through the selection of the best solution among a finite number of possible solutions. It is a particularly attractive technique to solve fuzzy optimisation problems because it allows finding close to optimal solutions, without big computational effort, which in a fuzzy environment is usually good-enough. In this context, we present the results for a set of problems, selected with the purpose of testing the SA algorithm performance. The problems tested were formulated following a complete fuzzification method proposed by Ribeiro and Moura-Pires (1999). These examples show the flexibility and adaptability of the fuzzy method as well as the SA algorithm, for the resolution of fuzzy linear optimisation problems.

Resumo

O algoritmo *Simulated Annealing* (*SA*) é uma ferramenta adequada à resolução de problemas de optimização linear, de natureza difusa, através da selecção da melhor solução de entre um número finito de soluções possíveis. Trata-se de uma técnica particularmente atractiva para resolver problemas de optimização difusa pois permite encontrar soluções próximas da óptima, à custa de um esforço computacional geralmente pouco exagerado. Neste contexto, apresentam-se os resultados para um conjunto de casos seleccionados, com a finalidade de testar o algoritmo de *SA* implementado. Uma vez que se pretende testar problemas de optimização difusos, usou-se um método de formulação proposto por Ribeiro e Moura-Pires (1999). Estes casos pretendem mostrar a flexibilidade e adaptabilidade do método difuso utilizado e do algoritmo de resolução *SA*, em problemas optimização linear difusos.

Keywords
Mathematical Programming, Fuzzy Optimisation, Fuzzy Sets, Simulated Annealing.

1 Introdução

Um Problema de Optimização Difuso pode ser entendido como um problema de optimização clássico transformado de modo a incluir alguma forma de "fuzificação" (do Inglês *fuzzification*), isto é, de flexibilização de modo a permitir uma tomada de decisão mais alargada e abrangente e, consequentemente, mais flexível e adequada em determinados cenários.

Em geral, os problemas de optimização visam a maximização ou a minimização de um objectivo simples (único) ou composto (múltiplo) enquanto satisfazem as restrições do problema, que representam as condições do modelo. O principal objectivo da abordagem difusa utilizada neste tipo de problemas é o de encontrar a solução ou a alternativa de decisão que satisfaz "melhor" uma determinada situação específica, que ocorre num ambiente difuso [9].

Subjacente a este trabalho está o desenvolvimento de uma aplicação informática que implementa um algoritmo de resolução do tipo *Simulated Annealing* (*SA*) pois é uma ferramenta adequada à resolução de problemas de optimização linear difusa, através da selecção da melhor solução de entre um número finito de soluções possíveis. Trata-se de uma técnica bastante atractiva para resolver problemas de optimização difusa pois permite encontrar soluções próximas da óptima, à custa de um esforço computacional geralmente não muito grande e é relativamente simples de implementar e manipular. No contexto deste trabalho um dos objectivos principais consiste portanto em mostrar a utilidade ou conveniência desta ferramenta para a resolução de problemas de optimização linear difusos e a principal medida de avaliação do seu desempenho consiste no valor que se obtém para a função objectivo, perante uma determinada situação de compromisso, subjacente a uma certa necessidade de violação do nível de satisfação das condições que caracterizam o problema em causa. Além disso também se introduziu na aplicação a possibilidade de proceder a uma análise do tempo de resposta do algoritmo *SA* em função do tipo de problema "fuzificado" (do Inglês *fuzzified*) considerado e far-se-á uma

análise comparativa destes valores, face aos resultados obtidos para os diversos tipos de problemas considerados, expressos em termos de valor da função objectivo e do correspondente nível de satisfação das condições do problema, delimitado por um determinado valor de *threshold* imposto no problema.

Adicionalmente, foi introduzida a possibilidade de o utilizador seleccionar uma "semente", para a geração dos números aleatórios, permitindo repetir experiências, a fim de possibilitar uma análise comparativa dos resultados obtidos para diferentes execuções do programa, com determinadas variações na formulação do problema. Além disso, também se inclui uma ligeira adaptação do algoritmo original, de modo a permitir a resolução de problemas de programação inteira.

O objectivo último da implementação deste algoritmo consiste não só em fornecer um resultado para problemas de optimização difusos, mas também servir como um meio de apoio à tomada de decisão, de modo a facultar um conjunto de soluções alternativas ao decisor, em tempo útil, indicando-lhe as consequências, em termos de vantagens e inconvenientes (compromissos), subjacentes a cada decisão, avaliados, essencialmente, pelo valor obtido para a F.O. e pelo nível de satisfação das condições do problema (*threshold*) e, deste modo, apoiá-lo na escolha da solução mais adequada a cada caso particular.

Neste trabalho pretende-se, também, ilustrar a flexibilidade de um método de optimização difusa proposto por Ribeiro e Moura-Pires [8,9] na formulação de diversos tipos de problemas de optimização linear difusos, que variam no tipo de "fuzificação" considerada e/ou nos desvios permitidos nas condições do problema e/ou no nível de satisfação (*threshold*) e/ou na dimensão do problema, além de outros parâmetros de controlo do algoritmo, que também podem variar, como se irá explicar mais adiante. O objectivo principal consiste em permitir apoiar a tomada de decisão na resolução deste tipo de problemas. Em particular, pretende-se mostrar que a principal vantagem da utilização deste método difuso reside na liberdade que o utilizador dispõe de flexibilizar qualquer problema específico de optimização. Sendo assim, a formulação difusa do problema pode variar desde um modelo simples em que, por exemplo, apenas se admitem variações nos parâmetros das restrições, até um modelo difuso global do problema, em que são permitidos desvios nos parâmetros e nos coeficientes das restrições e no valor e nos coeficientes da função objectivo.

A fim de expor estes assuntos achou-se conveniente estruturar este artigo do seguinte modo. No ponto 2 faz-se uma breve introdução à Tomada de Decisão Difusa, incluindo uma descrição de funções de pertença comuns, o modelo *Maxmin* de Belman e Zadeh [2], o modelo de optimização difusa de Zimmermann [11] e o método de "fuzificação" completo proposto por Ribeiro e Moura-Pires [8]. No ponto 3 descreve-se o algoritmo *Simulated Annealing (SA*) e apresentam-se os detalhes de implementação do algoritmo. No ponto 4 apresentam-se as formulações *crisp* e "fuzificadas" de um conjunto de casos que irão ser analisados (cobrindo um total de 84 casos de problemas difusos). Esta secção inclui a "fuzificação" do problema através da flexibilização que resulta da combinação de diversos tipos de "fuzificação", em termos de parâmetros das restrições, dos coeficientes da função objectivo e das restrições e do valor da função objectivo (meta). O ponto 5 apresenta os resultados para os casos considerados, resolvidos através do algoritmo de *SA* implementado e efectua uma análise destes, para as diferentes formulações consideradas. Finalmente, no ponto 6 extraem-se algumas conclusões e apresentam-se algumas intenções em termos de trabalho futuro.

2 Tomada de Decisão Difusa

Em geral poderá dizer-se que a Tomada de Decisão Difusa integra dois ramos fundamentais que são o "Multi-atributo Difuso" e a "Optimização Difusa" [6,11]. Neste trabalho aborda-se a problemática da Optimização Difusa quando tanto os coeficientes como os recursos e metas são difusos [8].

A tomada de decisão difusa recorre à Teoria de Conjuntos Difusos, desenvolvida por Zadeh [10]. Bundy [4] define a Teoria de Conjuntos Difusos como uma extensão à teoria de conjuntos convencional em que o grau de pertença para um elemento num conjunto toma um valor algures no intervalo [0,1], em vez de apenas 0 ou 1. Esta teoria foi desenvolvida no intuito de tornar tratável a complexidade subjacente a descrições de processos subjectivos ou mal entendidos.

Segundo Bellman [2] a tomada de decisão num ambiente difuso é considerada como um processo de decisão em que as metas a atingir e/ ou as restrições, mas não necessariamente o sistema sob controlo, são de natureza difusa. Isto significa que as metas e/ ou as restrições constituem classes de alternativas cujas fronteiras não estão definidas com precisão.

As metas e as restrições podem assim ser definidas de um modo adequado na forma de conjuntos difusos, no espaço de alternativas. Um conjunto difuso é uma classe de objectos em que não existe uma fronteira bem definida entre os objectos que pertencem à classe e os que não pertencem. Uma definição mais precisa poderá ser como se segue [2]:

Seja X={x} uma classe de objectos (pontos) representados genericamente por x. Então um conjunto difuso A em X é um conjunto de pares ordenados

$$A = \{(x, \mu_A(x))\},\ x \in X$$

onde $\mu_A(x)$ é designado por grau de pertença de x em A, e μ_A: $X \rightarrow M$ é a função de X para um espaço M, designado de espaço de pertença. Geralmente assume-se que M pertence ao intervalo [0, 1], em que 0 e 1 representam, respectivamente,

os graus ou níveis de pertença da lógica Booleana clássica. Sendo assim, assume-se que um conjunto difuso A pode ser definido com precisão, associando a cada objecto x um valor entre 0 e 1, que representa o seu grau de pertença em A.

2.1 Optimização Difusa

Os problemas tradicionais de optimização consistem em maximizar ou minimizar uma determinada função objectivo, sujeita a um conjunto de restrições, que exprimem, por exemplo, a limitação de recursos. O objectivo consiste então em maximizar ou minimizar a função objectivo, satisfazendo as restrições do problema.

$$\text{Max } f(x) \text{ sujeita a } x \in X$$

O objectivo principal da optimização difusa é o de encontrar a "melhor" solução (alternativa de decisão) na presença de informação incompleta, isto é, informação imprecisa e/ ou a existência de limites vagos na informação. Existem muitas formas de imprecisão, quando se manipulam ou abordam problemas de optimização difusa. Nomeadamente, coeficientes das variáveis que não se conhecem com precisão (por exemplo, "tempos de produção de cerca de uma hora, para a montagem de uma peça") ao nível de satisfação de restrições, que têm limites imprecisos (por exemplo, "o tempo total de produção disponível deverá rondar as 100 horas"). Um desafio proposto actualmente é o de permitir a construção de modelos que permitem utilizar e interpretar uma linguagem vaga e imprecisa, que possam ser traduzidos em métodos quantitativos difusos [9].

Um problema de optimização linear clássico pode ser formalizado como,

$$\begin{aligned} \max \quad & Z = C'x \\ & Ax \le B \\ & x \ge 0 \end{aligned}$$

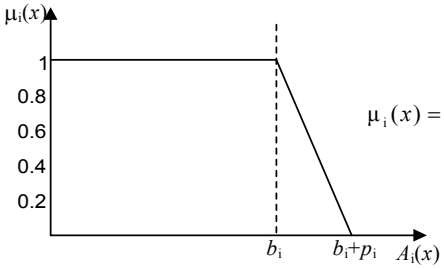
A versão "fuzificada" deste problema é geralmente formalizada como,

$$\begin{aligned} \max/\min \quad & \tilde{Z} = \tilde{C}\, x \\ & \tilde{A}\, x \{\ge, \le, =\} \tilde{B} \\ & x \ge 0 \end{aligned}$$

onde $\tilde{Z}$ representa uma meta difusa, $\tilde{C}$ é o vector de custos difusos, $\tilde{A}$ é a matriz que contém os coeficientes difusos do(s) objectivo(s) e das restrições e $\tilde{B}$ é o correspondente vector dos limites (parâmetros) do(s) objectivo(s) e dos recursos. Por vezes usa-se como notação um til em cima do sinal da equação/ inequação.

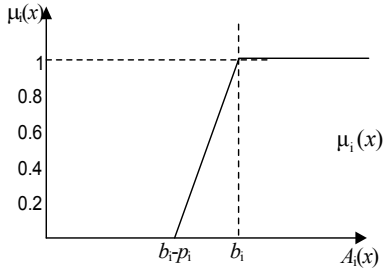
As primeiras propostas de "fuzificação" do problema de optimização apenas consideravam a "fuzificação" dos recursos (restrições) e de metas para os objectivos, como veremos nas próximas secções. Será no entanto de notar que existem vários métodos de "fuzificação" propostos na literatura, tanto para os recursos e metas, como para os coeficientes (ver detalhes em [5]). Neste artigo apenas focamos, em detalhe, o método de "fuzificação" completo proposto por Ribeiro e Moura-Pires [8], pois trata-se de um abordagem flexível e adaptável a problemas lineares e não lineares, conforme descrito na secção 2.4.

Um tipo de função de pertença, bastante utilizadas para representar os desvios aceites para os objectivos (metas), restrições e/ou coeficientes difusos são as funções triangulares. Com estas funções podemos representar facilmente qualquer tipo de "fuzificação" do problema de optimização difuso. Considerando um desvio p_i para flexibilizar as restrições e/ou para as metas de objectivos, os três tipos de funções mais comuns para representar equações de menor ou igual, maior ou igual e apenas igual, são:



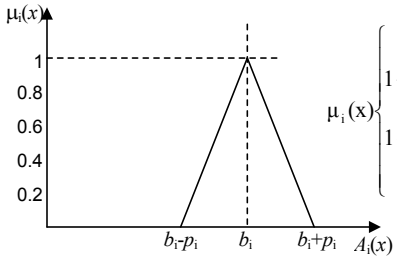
$$\mu_i(x) = \begin{cases} 0 & se \quad A_i(x) > b_i + p_i \\ 1 - \dfrac{A_i(x) - b_i}{p_i} & se \quad b_i < A_i(x) \le b_i + p_i \\ 1 & se \quad A_i(x) \le b_i \end{cases}$$

Figura 2.1 – Função de pertença para restrições do tipo «menor ou igual».



$$\mu_i(x) = \begin{cases} 0 & se \quad A_i(x) < b_i - p_i \\ 1 + \dfrac{A_i(x) - b_i}{p_i} & se \quad b_i - p_i \le A_i(x) < b_i \\ 1 & se \quad A_i(x) \ge b_i \end{cases}$$

Figura 2.2 – Função de pertença para restrições do tipo «maior ou igual».



$$\mu_i(x) \begin{cases} 0 & se \quad A_i(x) < b_i - p_i \\ 1 + \dfrac{A_i(x) - b_i}{p_i} & se \quad b_i - p_i < A_i(x) \le b_i \\ 1 - \dfrac{A_i(x) - b_i}{p_i} & se \quad b_i < A_i(x) < b_i + p_i \\ 0 & se \quad A_i(x) \ge b_i + p_i \end{cases}$$

Figura 2.3 – Função de pertença para restrições do tipo «igual».

A função representada na Figura 2.3 também pode ser usada para representar coeficientes difusos, como por exemplo "o custo por unidade x é de cerca de 200 escudos". Ou seja, a preferência do decisor é de 200 escudos mas são permitidos desvios à esquerda e à direita desta preferência. Por vezes esta representação triangular também serve para representar números difusos [8].

Note-se que o método de Zimmerman (secção 2.3) usa as funções de maior ou igual ou menor ou igual, apresentadas nas Figuras 2.1 e 2.2, para definir tanto as metas dos objectivos como as restrições de menor ou maior, respectivamente. É com base nestas funções que a transformação matemática proposta por Zimmerman [11] é possível e mantém a linearidade no problema transformado.

Quando se utilizam algoritmos independentes do problema para a sua resolução, como por exemplo o *SA*, qualquer outra função pode ser utilizada, nomeadamente uma função do tipo sinusoidal, ou trapezoidal, uma vez que não existem problemas de linearidade [9].

2.2 Modelo Simétrico

O primeiro método, *Maxmin*, para resolver problemas de decisão em ambiente difuso foi proposto por Bellman e Zadeh [2]. Estes autores consideram que a decisão sobre cada alternativa resulta da intersecção dos objectivos e das restrições (min) e que a melhor decisão é dada pela união das decisões sobre cada alternativa (max).

Uma decisão difusa pode então ser vista como a intersecção de metas e restrições o que significa tratar-se dum modelo simétrico. Uma decisão difusa de maximização é definida como um ponto no espaço de alternativas, no qual a função de pertença ou de membro de uma decisão difusa atinge o seu valor máximo [2]. Este modelo pode ser formalizado pela expressão matemática seguinte:

$$\mu_{D^*}(x) = \max \quad \min_{i=1}^{m+k} \mu_i(x), \forall x \in X$$

onde $\mu_{D^*}(x)$ é a decisão "óptima".

Grande parte da tomada de decisão ocorre num ambiente em que os objectivos, as restrições e as consequências de possíveis acções não são conhecidos com precisão. Para lidar quantitativamente com imprecisão torna-se por vezes necessário recorrer a ferramentas providenciadas pela teoria dos conjuntos difusos, de controlo e de informação [2].

Voltando ao conceito de decisão, observa-se que, intuitivamente, uma decisão é basicamente uma escolha ou um conjunto de escolhas alternativas efectuadas a partir de alternativas disponíveis. Esta ideia pode ser formalizada através da definição seguinte [2]: assumindo que existem metas difusas M e restrições difusas R, num espaço de alternativas X, então, M e R combinam-se para formar um conjunto difuso resultante da intersecção de M e R. Simbolicamente,

$$D = M \cap R$$

e, correspondentemente, $\mu_D = \mu_M \wedge \mu_R$. O "óptimo" é obtido pela escolha do melhor D para cada alternativa X (max de X).

No modelo de Bellman e Zadeh [2], não existe, portanto, diferença entre objectivos e restrições, no modelo do problema, assim como não existe diferença entre objectivos simples e múltiplos (compostos). Com o modelo simétrico, um problema de programação matemática torna-se num problema de satisfação de restrições, onde uma decisão é uma confluência de restrições propriamente ditas e objectivo(s).

2.3 Modelo Simétrico para Programação Linear

A primeira proposta de programação linear difusa deve-se a Zimmermann em 1976 [11]. Este autor considera que existe flexibilidade para a violação das restrições e objectivos e baseia a sua proposta no modelo simétrico de Bellman e Zadeh (isto é, não existe diferença entre restrições e objectivos).

Considerando a formalização clássica do problema de optimização, conforme definido na secção 2.1,

$$\begin{aligned} \max \quad & Z = C'x \\ & Ax \le b \\ & x \ge 0 \end{aligned}$$

e usando o modelo simétrico [2] podemos formalizar o problema como:

$$R\ (x) \le \widetilde{B} \quad \text{onde} \begin{pmatrix} -C \\ A \end{pmatrix} = R \ \text{e} \begin{pmatrix} -Z \\ b \end{pmatrix} = B$$

$$\text{e } D^*(x) = \max \min \left(1 - \left(\frac{R_i - B_i}{p_i} \right) \right) \text{ para } x \ge 0$$

Considerando as funções de pertença (Figuras 2.1 e 2.2) para as tolerâncias p_i , definidas à *priori* pelo decisor e que representam os níveis de violação do objectivo e das restrições, Zimmermann [11] propõe a seguinte transformação linear,

$$\begin{aligned} & \text{Max } \lambda \\ & \text{s.a.} \\ & \qquad \lambda p_i + R_i(x) \le p_i + B_i \\ & \qquad x \ge 0,\ i = 1,\ldots,\ m+1 \end{aligned}$$

em que λ representa a medida de satisfação de restrições difusas e onde p_i são os desvios dos valores *crisp*[1].

Note-se que, com esta transformação, o problema pode ser resolvido com um algoritmo simples como o *Simplex*, o que representa uma vantagem deste método. No entanto, este modelo não é compatível com a "fuzificação" de coeficientes tanto na F.O. como nas restrições e esta é a razão de ser necessário recorrer a outros métodos.

2.4 Método de "Fuzificação"/ Flexibilização Completa

Em geral a "fuzificação" completa do modelo de programação linear, de optimização difusa, inclui seis formas de imprecisão, como se apresenta a seguir [9]:

Caso 1 : Imprecisão nos limites das restrições. Isto implica a "fuzificação" dos limites das desigualdades (por exemplo, "o tempo total para pintar um apartamento deverá ser consideravelmente inferior a 100 horas").

Caso 2: É imposta uma meta difusa para a função objectivo. Isto implica essencialmente a "fuzificação" da função de utilidade (por exemplo, "o custo total para o projecto deverá ser mantido consideravelmente abaixo de cem mil euros").

Caso 3: Imprecisão composta: que implica a combinação de imprecisões dos tipos anteriores.

Caso 4: Os coeficientes das variáveis das restrições não são conhecidos com precisão. Isto significa que os coeficientes são números difusos (por exemplo, "o custo por hora de produção de uma camisa x, é cerca de 5 euros").

Caso 5: Os coeficientes das variáveis da função objectivo não são conhecidos com precisão. Isto significa que esses coeficientes são números difusos, tal como em 4 (por exemplo, "o preço de venda do produto x é cerca de 10 euros por item").

Caso 6: Todas as combinações possíveis de incerteza.

Em resumo, o problema de optimização difuso com coeficientes difusos e parâmetros difusos pode ser expresso matematicamente por:

[1] Entenda-se por valor *crisp* todo aquele valor normal, que está associado a uma a formalização clássica do problema de optimização linear, isto é, um valor não "fuzificado".

$$\max/\min \quad Z = \tilde{C}\, x$$
$$\tilde{A}\, x \{\tilde{\geq}, \tilde{\leq}, \cong\}\, B$$
$$x \geq 0$$

onde $\tilde{C}$ é o vector de custos difusos, $\tilde{A}$ é a matriz que contém os coeficientes difusos do(s) objectivo(s) e das restrições e $\tilde{\geq}, \tilde{\leq}, \cong$ representa o sinal correspondente à "fuzificação" dos limites (parâmetros) do(s) objectivo(s) e/ ou dos recursos (cuja informação, alternativamente, se pode representar por $\tilde{B}$).

Vários autores propuseram formalizações para resolver o caso da "fuzificação" completa, como por exemplo Carlsson e Korhonen [citados em 5] e Ribeiro e Moura-Pires [8]. Neste trabalho apenas discutimos em detalhe o método de Ribeiro & Moura-Pires pois é flexível e facilmente adaptável a problemas com objectivo único ou múltiplo, assim como para problemas lineares e não-lineares.

Método de "fuzificação" completa (flexível) proposto por Ribeiro & Moura-Pires

Ribeiro e Moura-Pires propuseram uma "extensão" aos métodos de resolução de problemas de optimização difusos existentes, por forma a incluir coeficientes difusos tanto na função objectivo como nas restrições. Todo o processo é designado de "abordagem flexível" para problemas de optimização difusos [7,9] e a sua formalização é:

$$\max_x \; Z = \sum_{k=1}^{K} \tilde{e}_k \cdot x_k$$
$$\sum_{k=1}^{K} \tilde{a}_{ik} \cdot x_k = \tilde{b}_i, \; i = 1,..., N$$
$$x \geq 0$$

que é convertida no seguinte sistema de equações não-lineares:

$$\max_x \; Z = \sum_{k=1}^{K} w_k \cdot x_k$$
$$\max_{y,w} \; (\mu_{aik}, \mu_{bi}, \mu_{ck})$$
$$\sum_{k=1}^{K} y_{i,k} \cdot x_k \{=, \leq, \geq\} \tilde{b}_i$$
$$y_{i,k} = \tilde{a}_{i,k}$$
$$w_k = \tilde{c}_k$$
$$\text{onde} \quad k = 1,..., K \text{ e } i = 1,..., N \text{ e } x, y, w \geq 0$$

Note-se que todos os parâmetros difusos são apresentados com um til por cima e que as funções de pertença usadas são idênticas à representação constante nas Figuras 2.1, 2.2 e 2.3, isto é, são usadas funções triangulares. Note-se que quaisquer outras funções de pertença poderiam ser utilizadas.

Neste processo de flexibilização c_k e a_{ik} difusos tornam-se variáveis do processo. Poderá mesmo acontecer que um destes coeficientes seja dependente de outro c_k ou a_{ik}. Estas dependências podem ser facilmente incorporadas no modelo expressando os relacionamentos existentes. Note-se que com este novo método apenas as variáveis independentes são variáveis do problema de optimização.

O método de resolução proposto nesta abordagem assenta em 2 passos fundamentais:

1º) Encontrar o valor de x, y e w, variáveis independentes que maximizam o μ (*Niu*) agregado das restrições.

2º) Encontrar o valor óptimo de Z que satisfaz todas as restrições com um α-*cut*[1]$= \alpha_{max}$.

Este método permite manipular qualquer tipo de problema de optimização difuso linear ou não linear. Outra vantagem é a possibilidade de resolver problemas *crisp* que não têm solução, característica que é facultada através do seu funcionamento num ambiente difuso e pela inclusão de imprecisão quando os coeficientes ou os parâmetros não são conhecidos com precisão.

Em conclusão poderá dizer-se que este modelo de "fuzificação" aumenta a possibilidade de estabelecimento de compromissos entre satisfação de restrições e o valor para a meta a atingir, ao dispor do decisor e, deste modo, permite uma tomada de decisão mais flexível.

[1] Um α-*cut* é um valor de limiar que define um limite mínimo para o nível de satisfação das restrições do problema considerado.

3 Resolução de Problemas de Optimização

A análise da complexidade indica se é possível resolver um determinado problema de optimização e se é possível encontrar uma solução óptima, num intervalo de tempo superiormente limitado por um polinómio em função do comprimento de entrada (dados) do problema, ou seja, em tempo polinomial. Um algoritmo de complexidade polinomial é um algoritmo cuja função de complexidade temporal é O(p(k)), onde p é um polinómio qualquer e k é o comprimento de entrada de uma instância do problema. Cada algoritmo cuja função de complexidade temporal não pode ser limitada deste modo será designado algoritmo de tempo exponencial.

A classe de problemas P consiste em todos os problemas de decisão que poderão ser resolvidos através da máquina determinística de *Turing,* um modelo abstracto de computação, em tempo limitado superiormente por um polinómio do comprimento de entrada [3]. Sabe-se que $P \subseteq NP$. A classe de problemas *NP*, de problemas de decisão, consiste em todos os problemas de decisão que poderão ser resolvidos em tempo polinomial por uma máquina não determinística de *Turing*. Esta classe de problemas inclui duas subclasses de problemas que são os problemas do tipo *NP-Hard* e *NP-Complete* [3].

De entre a enorme variedade de algoritmos de optimização é possível identificar um conjunto de procedimentos que podem ser agrupados em três categorias fundamentais [3]:

- Procedimentos matemáticos, exactos ou de aproximação;
- Procedimentos baseados em Simulação;
- Procedimentos heurísticos diversos.

Os procedimentos matemáticos exactos correspondem a algoritmos de resolução exactos, que têm por objectivo encontrar a solução óptima. Três classes fundamentais destes algoritmos são os Algoritmos Exactos de Programação Linear, de Programação Dinâmica e os Algoritmos de "Ramificação e Limite" exactos ou puros (*Branch and Bound - B&B*). Estes algoritmos, sendo exaustivos, não são adequados à resolução de problemas de grandes dimensões ou "difíceis". Sendo assim, existem, alternativamente, os algoritmos ou procedimentos heurísticos, que são algoritmos mais apropriados para a resolução de problemas "difíceis". Embora estes métodos de resolução não garantam a obtenção de soluções óptimas, são normalmente mais expeditos e permitem, geralmente, obter soluções boas ou pelo menos aceitáveis, podendo mesmo chegar à solução óptima. Grande parte destes algoritmos são relativamente recentes e têm sido cada vez mais explorados, como é o caso dos Algoritmos de Pesquisa na Vizinhança ou de Pesquisa Local Extendidos (*Extended Neighborhood Search*), que incluem *Simulated Annealing*, Algoritmos de Pesquisa Tabu assim como Algoritmos Genéticos. Existem ainda Algoritmos de Pesquisa em Feixe (*Beam Search Methods*), Algoritmos Aproximados de Programação Dinâmica e outros Algoritmos Aproximados de Ramificação e Limite, baseados em técnicas *B&B*, entre outros, nomeadamente, um conjunto de *Bottleneck Methods* [3].

Muitos dos problemas de optimização são do tipo *NP-Complete*. Geralmente acredita-se que os problemas deste tipo não são resolúveis em tempo polinomial. Consequentemente, existe muito interesse em usar algoritmos heurísticos ou de aproximação, que permitem encontrar uma boa solução, em tempo de execução razoável ou aceitável, o que motivou a realização deste estudo. Este trabalho centra-se num dos algoritmos pertencentes à classe dos algoritmos de pesquisa local ou de pesquisa na vizinhança, mais precisamente, no *Simulated Annealing*. Existem vários esquemas de pesquisa, mas todos têm a mesma característica de possuir uma função de vizinhança subjacente, que é utilizada para guiar a pesquisa até uma boa solução.

O tratamento das vizinhanças depende do problema considerado e encontrar funções de vizinhança eficientes, que conduzem a óptimos locais de elevada qualidade pode ser visto como um dos desafios colocados à pesquisa local em geral. Até agora não existem ainda regras gerais e cada caso terá de ser resolvido em particular. Estes problemas de optimização podem ainda ser problemas de natureza difusa, como é o caso dos problemas que aqui se pretendem analisar.

3.1 Algoritmo de *Simulated Annealing*

Um algoritmo de *Simulated Annealing* (*SA*) pertence a uma classe de algoritmos de pesquisa local que também são conhecidos sob a designação genérica de "algoritmos de limiar" (do inglês, *Threshold Algorithms*). Estes algoritmos desempenham um papel importante na pesquisa local, por duas razões; em primeiro lugar, porque têm sido bem sucedidos, quando aplicados a um vasto conjunto de problemas práticos, o que lhes atribuiu uma considerável afirmação entre os seus utilizadores. Em segundo lugar, alguns desses algoritmos, tal como é o caso do *SA*, têm uma componente estocástica, que facilita uma análise teórica da sua convergência assimptótica e isto tornou-os bastante populares entre os matemáticos [1].

O algoritmo de *Simulated Annealing* foi originalmente proposto por Kirkpatrick e outros, em 1983 e por Cerny, em 1988, por analogia com o processo de fundição (*annealing*) de um sólido (citado em [1]). O objectivo de um algoritmo desta natureza é encontrar a melhor solução de entre um número finito de soluções possíveis. A técnica de *SA* é uma técnica particularmente atractiva pois permite encontrar soluções próximas da óptima, à custa de um esforço computacional geralmente pouco exagerado. É de notar que neste tipo de algoritmo, não é possível saber se a melhor

solução encontrada, é o óptimo global. Esta característica restringe a sua utilização a casos em que um bom óptimo local é aceitável [3].

Em procedimentos do tipo *SA*, as sequências das soluções não tendem linearmente para um óptimo local, como acontece noutras técnicas de pesquisa local. Sendo assim, verifica-se que as soluções traçam um percurso ou trajecto variável, através de um conjunto S, de soluções possíveis e este percurso tende a ser guiado numa direcção "favorável".

Regra geral, poderá dizer-se que o *SA* consiste num procedimento fiável para usar em situações em que o conhecimento é escasso ou se aparenta difícil de aplicar algoritmicamente. Mesmo para dar soluções a problemas complexos, esta técnica é relativamente fácil de implementar e normalmente executa um procedimento do tipo *hill-climbing* com múltiplos recomeços. Tipicamente, esta técnica gera um caminho "Markoviano", em que o sucessor de um ponto actual é escolhido estocasticamente, com uma probabilidade que depende apenas do ponto actual e não da história prévia da busca. A propriedade "Markoviana" é uma bênção misturada, ou seja, permite resultados heurísticos, que são bastante bons em muitos casos, mas torna a análise teórica do algoritmo difícil. O algoritmo de *SA* é claramente desapropriado para a resolução óptima de problemas de optimização.

Em resumo, o algoritmo de *Simulated Anne*aling (*SA*) é um algoritmo estocástico com uma analogia "física" de "fusão" do sistema a ser optimizado (como um sólido) usando "temperaturas elevadas" e posteriormente prosseguindo através de uma redução progressiva e suave da temperatura, até que o sistema "cristalize" e não ocorram mais alterações. O processo de fusão pode ser visto como a busca pela melhor solução. Extensões a este tipo de algoritmo permitem também resolver problemas de optimização difusa pois não são dependentes da linearidade da função objectivo ou das restrições [9]. Sendo assim, poderão referir-se as seguintes vantagens. Ao usar uma implementação deste algoritmo em que os limiares (α-*cuts*) podem ser definidos, o decisor poderá definir o grau de satisfação (μ_{Ak} e/ ou μ_{aij}) para o parâmetro de cada restrição e /ou de cada coeficiente (da(s) função(ções) objectivo e/ ou da(s) restrição(ções)) do problema. Acresce que esta técnica de resolução permite tratar problemas de optimização difusos, que podem ser abordados, de um modo controlado, através da consideração da possibilidade de existência de desvios (violações) em relação aos valores normais (*crisp*). Isto é, o algoritmo *SA* permite lidar com qualquer um dos tipos de "fuzificação" referidos no ponto 2.4.

3.2 Implementação do *Simulated Annealing*

Como referido, o algoritmo *SA* é um algoritmo usado para problemas de optimização onde a função objectivo (F.O.) corresponde à energia dos estados de um sólido. O algoritmo *SA* requer a definição de uma estrutura de vizinhança, assim como os parâmetros para a programação do arrefecimento. Um parâmetro de temperatura permite distinguir entre alterações profundas ou ligeiras na função objectivo. Alterações drásticas ocorrem a elevadas temperaturas e modificações pequenas a temperaturas baixas. Trata-se de um processo evolucionário, que se move em pequenos passos, de um estado para outro e existe o problema de ficar preso num óptimo local, o que constitui uma consequência deste tipo de algoritmo [9]. De acordo com [9], os quatro requisitos básicos para o utilizar o algoritmo *SA* em problemas de optimização combinatorial são:

- descrição concisa do problema;
- geração aleatória das alterações de uma configuração para outra;
- uma F.O. contendo a função de utilidade dos compromissos;
- uma definição do estado inicial, do número de iterações a serem executadas para cada temperatura e o seu esquema de *annealing*.

O pseudo código do algoritmo de *SA* utilizado (baseado na implementação apresentada em [9]) é o que se apresenta a seguir:

Inicializar o contador de mudança de temperatura $t := 0$;

Seleccionar a temperatura inicial $T > 0$;

Seleccionar *threshold* > 0;

Inicializar a melhor solução *Zm* := solução inicial *crisp*;

Inicializar o Gerador de números aleatórios *Seed* >= 0;

Gerar Coef. das restrições *fuzzy* na vizinhança dos coef. *crisp*;

Para k:=1 até *Nr* fazer $\mu(k)$:= valor de pertença da restrição $A_k(x)$;

Niu1 = agregação($\mu(1)$, ..., $\mu(Nr)$);

```
Repetir
    Colocar contador de repetição n := 0;
    Repetir
        Gerar estado y na vizinhança de x;
        Gerar Coef. das restrições fuzzy na vizinhança dos coef. crisp;
        k := 1;
        Repetir
            μ(k) := valor de pertença da restrição A_k(x);
            k++;
        ate k >Nr ou μ(k-1)<=threshold;
        se μ(k-1)>=threshold então
            Niu2 = agregação(μ(1), ..., μ(Nr));
            δ := Niu2 - Niu1;
            se δ>0 então x:= y;   Niu1:=Niu2;
                    senão  se random(0,1)< exp(δ/T)
                            então x:=y; Niu1:=Niu2;
            se Zm  melhor que Zfuzzy então
                Guardar melhor óptimo intermédio
        n := n + 1;
    até  n = N(t) ;
    t := t + 1;
    T := T(t);
Até se verificar o critério de paragem;
fim;
```

A variável k é um contador para o número de restrições. É usada para calcular o grau de pertença de cada restrição μ(k). *Niu* representa a agregação (intersecção) dos graus de pertença das restrições difusas e representa o menor nível de satisfação das restrições (ou seja, o melhor possível obtido) . Qualquer operador de agregação poderia ser utilizado, mas neste trabalho foi utilizado o operador *t-norm min*. A variável δ exprime a diferença entre o valor prévio do *Niu* e o novo valor obtido (*Niu* corrente) e é utilizada para escolher o máximo *Niu*, ou seja, o melhor nível de satisfação das restrições. Esta variável representa a diferença de energia entre o estado anterior e o novo estado. A função de probabilidade de movimentação, para um estado de melhor energia, (o objectivo) é dada pela exponencial de δ dividido pelo parâmetro de controlo T. Quanto menor a temperatura T, menos provável será a ocorrência de alguma alteração. O N(t) representa o número de soluções vizinhas geradas e, consequentemente, o número de soluções possíveis. O T(t) é a função de decréscimo da temperatura.

Foram ainda efectuadas as seguintes escolhas para a implementação *SA*, baseadas no trabalho apresentado em [14]. Na implementação do algoritmo *SA* tornou-se necessário especificar os seguintes parâmetros: como gerar um estado y, vizinho de x; que função de agregação (*Niu*) usar; o número de vizinhos gerados, N(t); a função de decréscimo da temperatura, T(t) e finalmente o critério de paragem.

A escolha de como gerar um estado y, vizinho de x, é efectuada através da definição de um novo estado, que é um ponto aleatório y, em que a distância desse novo ponto ao ponto actual z é aleatória e inferior a ρ(t), definida por:

$$\rho(t) = \begin{cases} 1 & se \quad t < 100 \\ 2 & se \quad 100 \le t < 150 \\ 3 & se \quad 150 \le t < 250 \\ 5 & se \quad 250 \le t < 350 \\ 15 & se \quad t \ge 350 \end{cases}$$

Como referido a função de agregação é a intersecção de todos os valores de pertença das restrições e o operador usado para a intersecção é o *t-norm min*. A intersecção (dada pelo *Niu*) representa o "e" lógico, para indicar que todas as restrições têm de ser satisfeitas com um certo nível (*min*). O objectivo torna-se então o da maximização da minimização dos desvios das restrições.

Os pontos vizinhos são gerados através do uso de coordenadas polares, que garantem a aleatoriedade necessária à selecção dos novos pontos a considerar. Para *n* variáveis de decisão, tem-se que para Δx,

$$\Delta x_1=\rho\times\cos\theta_1 \qquad \theta_1 \in [0,2\pi]$$
$$\Delta x_2=\rho\times\text{sen}\theta_1\times\cos\theta_2 \qquad \theta_2 \in [0,\pi]$$
$$\ldots$$
$$\Delta x_{k-1}=\rho\times\text{sen}\theta_1\times\ldots\times\text{sen}\theta_{k-2}\times\cos\theta_{k-1} \qquad \theta_{k-1} \in [0,\pi]$$
$$\Delta x_k=\rho\times\text{sen}\theta_1\times\ldots\times\text{sen}\theta_{k-2}\times\text{sen}\theta_{k-1}\times\text{sen}\theta_k \qquad \theta_k \in [0,\pi]$$

O novo ponto, na vizinhança do anterior é então dado por:

$$y = x + \Delta x$$
$$(y_1,..., y_n) = (x_1,..., x_n) + (\Delta x_1,..., \Delta x_n)$$

Deve notar-se que, após a geração de um novo ponto na vizinhança é necessário testar se esse ponto (vector de valores, y) satisfaz as restrições do problema difuso e além disso, no caso de problemas do tipo inteiro, haverá a necessidade de proceder a um arredondamento dos componentes do vector y (ou a selecção ou adaptação das funções de pertença para estes casos), donde que:

$$y = x + \text{int}(\Delta x + 0.5)$$

A selecção do número de vizinhos a gerar N(t), segue a seguinte heurística:

$$N(t)\begin{cases}200 & se \quad t < 400\\ 250 & se \quad t \geq 400\end{cases}$$

Esta heurística tem em consideração o facto de que mais sucessores deverão ser gerados quando a temperatura T diminui, para ter mais opções para testar. A função de temperatura T(t) é uma função que usa um factor de decréscimo de 0.99. O critério de paragem do algoritmo é atingido quando a temperatura é inferior ou igual a 0.0001.

4 Problemas de Optimização Linear Testados

4.1 Formulação dos Problemas Lineares *Crisp*

Neste trabalho foram considerados dois problemas do tipo *crisp*, como problemas de base para a formulação de problemas "fuzificados". O primeiro problema ("*Crisp* 1") considera 10 restrições, desde a restrição 11 à restrição 20, e o segundo problema ("*Crisp* 2") contempla a totalidade das restrições (22), que se apresentam na Tabela 4.1.

Restrição	x_1	x_2	x_3	x_4	x_5	x_6	x_7	sinal	parâmetro
1	0.1	0.13	0.11	0.13	0.13	0.1	0.11	≤	500
2	0.77	1	0.91	1.11	1.25	0.83	0.77	≤	3000
3	0.91	0.83	0.91	0.91	0.91	0.83	0.77	≤	3600
4	0.08	0.09	0.09	0.08	0.1	0.1	0.09	≤	600
5	1	1.67	1.43	1.25	1.11	1.25	1	≤	6000
6	0	2.5	2.5	3.33	0	0	0	≤	3600
7	0	0.17	0.25	0.2	0	0	0	≤	500
8	0	0	0	0	2.86	0.2	0.25	≤	600
9	0.33	0.5	0.5	0.56	0.67	0.33	0.63	≤	1500
10	0.2	0.2	0.25	0.33	0.25	0.29	0.29	≤	1800
11	0.25	0	0.5	0	0.33	0	0	≤	600
12	0.25	0.29	0.29	0.25	0.2	0.25	0.33	≤	1800
13	0	0	1	0	0	0	0	≤	50
14	0	0	0	1	0	0	0	≤	60

15	0	0	0	0	0	1	0	≤	35
16	0	0	0	0	0	0	1	≤	45
17	1	1	1	1	1	1	1	≤	150
18	1	0	0	0	0	0	0	≥	10
19	0	1	0	0	0	0	0	≥	20
20	0	0	1	0	0	0	0	≥	20
21	0	0	0	1	0	0	0	≥	20
22	0	0	0	0	1	0	0	≥	10

$x_j \geq 0$ (j = 1, 2, ..., 7)

Tabela 4.1 – Restrições dos problemas *crisp*.

A função objectivo escolhida para teste é a seguinte.

Max $200\,x_1 + 300\,x_2 + 400\,x_3 + 500\,x_4 + 600\,x_5 + 600\,x_6 + 650\,x_7$

Ao problema "*Crisp* 1" corresponde um valor de F.O. de 78250 e ao problema "*Crisp* 2" um valor de 76250, para os seguintes valores das variáveis de decisão:

Problema	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Crisp1	10	20	20	0	55	0	45
Crisp2	10	20	20	20	10	25	45

Tabela 4.2 – Valores das variáveis de decisão para os problemas *crisp*.

4.2 "Fuzificação" do Problema

Nos problemas de optimização linear foram considerados os seguintes 7 casos de flexibilização:

Caso 1. "Fuzificar" os coeficientes da função objectivo e o valor da função objectivo F.O. (meta);

Caso 2. "Fuzificar" valor da função objectivo e o valor das restrições (parâmetros).

Caso 3. "Fuzificar" os coeficientes e o valor da F.O. e o valor das restrições.

Caso 4. "Fuzificar" o valor da F.O. e os coeficientes das restrições.

Caso 5. "Fuzificar" os coeficientes e o valor da F.O. e os coeficientes das restrições.

Caso 6. "Fuzificar" o valor da F.O. e os coeficientes e os valor das restrições.

Caso 7. "Fuzificar" os coeficientes e o valor da F.O. e os coeficientes e o valor das restrições.

Estas formas de flexibilização foram então aplicadas a cada uma das formulações *crisp* de base, referidas no ponto 4.1, perfazendo 14 problemas. Cada um destes problemas foi ainda submetido a uma análise de flexibilização com desvios permitidos de 10 e 15%, dando origem a 28 casos distintos. Estes 28 casos, por sua vez, foram ainda testados sob 3 cenários distintos, em termos de valor de limiar imposto no nível de satisfação das condições destes problemas (*threshold*), correspondentes a 30, 60 e 90%. Sendo assim, no total, foram testadas 84 situações distintas, com a finalidade de proceder a uma análise suficientemente detalhada e diversificada dos resultados obtidos pelo algoritmo de *SA* para os diferentes cenários considerados.

De acordo com o formalismo apresentado no ponto 2.4, quando num problema surgem coeficientes flexibilizados, tanto na F.O como nas restrições e/ou se considera a "fuzificação" de parâmetros das restrições (e/ou do valor da F.O - meta) estas flexibilizações serão denotadas, na correspondente formulação, através de um til, colocado por cima do respectivo coeficiente e/ou sinal da inequação, por exemplo, $\tilde{0.33}$ e $\tilde{\leq}$.

5 Resultados dos Problemas de Optimização

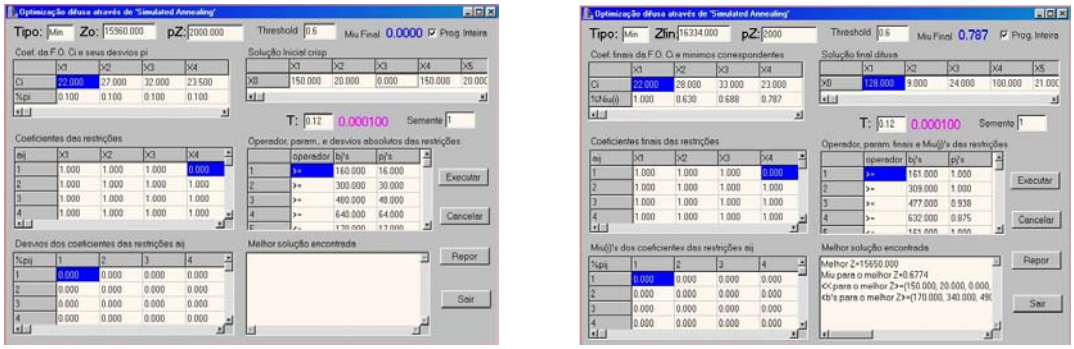
Em geral o processo para resolver um problema de optimização difuso consiste nos seguintes passos:

1. Dependendo do tipo de problema, formalizá-lo na forma de problema de programação linear ou na forma de um problema multi-objectivo, ou mesmo como um problema de programação não-linear.

2. Se se pretender "fuzificar" os objectivos, definir a meta para esses objectivos.
3. Definir a função de pertença para representar a "fuzificação" de qualquer restrição. (Ex: triangular, sinusoidal, trapezoidal, ou outra).
4. Definir limiares para o grau de aceitação de desvios (violações) na satisfação das restrições ou condições.
5. Definir o operador de agregação para combinar as restrições (e metas) como por exemplo uma "t-norma".
6. Resolver o problema com um algoritmo do tipo *SA*.

Note-se que existe a possibilidade de correr o algoritmo *SA* para diferentes valores de limiar (α-*cut* ou *threshold*) e o objectivo consiste em permitir uma comparação entre diferentes resultados obtidos, para um mesmo problema, a fim de analisar se é possível convergir para um determinado ponto ou zona de pontos óptimos ou para tentar extrair as conclusões que se apresentem mais promissoras para a resolução de um determinado problema ou, simplesmente, para ter uma noção de que existem diferentes alternativas de decisão, às quais correspondem determinadas situações de compromisso, em termos de optimização do valor da F.O. (Z) e do nível de satisfação global das condições do problema (*Niu*).

A Figura 5.1 permite uma visualização das janelas de diálogo que servem de interface para a introdução dos dados e para a apresentação dos resultados dos diversos problemas, respectivamente.



(a) (b)

Figura 5.1 – Janelas de diálogo da aplicação, para a introdução dos dados (a) e para a apresentação de resultados (b).

É ainda importante relembrar que os exemplos que se apresentam a seguir foram executados para valores de limiar (*threshold*) de 0.3, 0.6 e 0.9, para os diferentes casos de "fuzificação" abordados.

Relativamente aos casos de "fuzificação" considerados estes foram divididos em dois grupos fundamentais de problemas, A e B, que são os problemas em que se consideram, respectivamente, desvios ou violações de 10%, relativamente aos valores *crisp* (Tabela 5.1) e desvios de 15% (Tabela 5.2).

Também se achou conveniente utilizar uma "semente", no algoritmo, para a geração de números aleatórios, para permitir a geração dos mesmos valores, em diferentes execuções do programa, para um mesmo exemplo, a fim de ser possível comparar os diferentes resultados obtidos, face a pequenas variações nas condições de execução de cada problema. Nestes problemas utilizou-se uma semente de valor 1.

Além disso também se implementou um relógio, que permite avaliar o tempo de resposta (em segundos) aos vários problemas e que estão expressos pela variável "Tempo", nas tabelas abaixo.

As tabelas seguintes apresentam ainda o valor da F.O. (Z), para cada problema considerado e o correspondente nível de satisfação das condições do problema (*Niu*). Esta variável contempla desvios no valor das restrições, no valor da F.O. e nos correspondentes coeficientes e combinações destas situações, de acordo com os casos de "fuzificação" referidos. Será ainda de acrescentar que *Niu* representa o nível mínimo de violação das restrições, ou seja, a melhor satisfação possível encontrada para o valor da F.O.

As designações dos problemas contêm informação relativa ao tipo de caso de "fuzificação" de que se trata (casos 1 a 7) (Ex.: Caso**1**.yz, para um problema da classe dos problemas "Caso 1"), informação relativa ao problema *crisp* em que se baseia (Ex.: Casox.**1**Z), que corresponde ao problema "*Crisp* 1") e ainda informação relativa à classe de problemas em que se insere, relativamente aos desvios ou violações permitidas (Ex.: Casox.y**A**, para um problema pertencente à classe A, caracterizado por desvios de 10%).

Os resultados para os problemas da classe A foram os seguintes:

Desvios 10%	*Threshold*=0.3			*Threshold*=0.6			*Threshold*=0.9		
	Z	*Niu*	Tempo	Z	*Niu*	Tempo	Z	*Niu*	Tempo
Caso1.1A	82223.584	0.4059	297	80321.289	0.6050	648	78606.265	0.9059	2116
Caso1.2A	79775.478	0.3458	990	77901.232	0.6232	1402	76575.161	0.9676	2524
Caso2.1A	82935.475	0.3003	59	80623.626	0.6037	51	78796.001	0.9083	42
Caso2.2A	81520.987	0.3019	135	79040.269	0.6044	129	76889.279	0.9109	126
Caso3.1A	88864.288	0.3366	93	82535.396	0.6064	140	79276.940	0.9010	681
Caso3.2A	85833.658	0.3237	142	80753.384	0.6058	154	77348.281	0.9005	169
Caso4.1A	79496.303	0.3310	934	79748.248	0.6332	2686	78645.183	0.9013	33
Caso4.2A*	76163.606*	0.3385	193	76055.217*	0.6037	372	76104.835*	0.9005	2287
Caso5.1A	85689.700	0.3238	116	81218.748	0.6311	314	78967.500	0.9014	329
Caso5.2A	78994.391	0.3065	137	77604.922	0.6169	273	76433.001	0.9001	1267
Caso6.1A	82995.997	0.3240	182	81379.872	0.6000	240	78907.454	0.9001	850
Caso6.2A	82411.910	0.3001	210	79290.528	0.6012	255	77000.635	0.9016	631
Caso7.1A	91032.668	0.3127	179	84524.696	0.6031	277	79200.534	0.9004	2643
Caso7.2A	85582.212	0.3100	226	80973.784	0.6028	293	77079.792	0.9041	796

Tabela 5.1 – Resultados para os problemas da classe A.

Os resultados para os problemas da classe B foram os seguintes:

Desvios 15%	*Threshold*=0.3			*Threshold*=0.6			*Threshold*=0.9		
	Z	*Niu*	Tempo	Z	*Niu*	Tempo	Z	*Niu*	Tempo
Caso1.1B	84037.685	0.8449	161	81499.400	0.6270	95	78795.698	0.9738	1978
Caso1.2B	80953.229	0.3051	192	78999.003	0.6044	181	76709.386	0.9109	1671
Caso2.1B	86308.608	0.3215	114	82371.776	0.6046	160	78951.227	0.9034	373
Caso2.2B	85024.635	0.3404	135	80704.633	0.6200	129	77367.162	0.9003	124
Caso3.1B	89596.860	0.3077	83	86290.221	0.6000	108	79704.568	0.9003	422
Caso3.2B	88283.349	0.3397	139	83891.885	0.6025	147	77864.135	0.9008	153
Caso4.1B	83591.246	0.3238	86	80539.869	0.6050	83	78909.421	0.9024	2627
Caso4.2B*	76104.068*	0.3304	83	76044.617*	0.6120	130	76045.359*	0.9023	1178
Caso5.1B	88934.092	0.3185	75	82575.194	0.6022	377	79091.023	0.9031	3435
Caso5.2B	81857.123	0.3044	97	78794.372	0.6126	265	76377.843	0.9012	955
Caso6.1B	83719.261	0.3011	146	81379.492	0.6001	335	78972.459	0.9006	1121
Caso6.2B	86197.369	0.3022	187	79298.176	0.6002	218	77012.363	0.9000	524
Caso7.1B	89047.650	0.3035	157	84771.080	0.6084	452	79815.201	0.9000	1898
Caso7.2B	90734.469	0.3036	203	83884.405	0.6109	243	77649.240	0.9002	615

Tabela 5.2 – Resultados para os problemas da classe B.

Os problemas “Caso4.2A” e “Caso4.2B”, assinalados com ‘*’, nas Tabelas 5.1 e 5.2, correspondem aos únicos exemplos de problemas, de entre os que foram analisados, em que não se verifica uma melhoria no valor da F.O. (Z), como se pode observar nos resultados tabelados.

5.1 Análise de Resultados

As experiências, correspondentes aos 7 casos de "fuzificação" apresentados no ponto 4.2, foram executadas partindo de uma temperatura inicial de 0.12. Os diferentes tipos de violações ou desvios permitidos foram de 10% nos problemas do tipo A e de 15 % nos problemas do tipo B, isto tanto nos valores dos parâmetros das restrições (e da F.O.) como nos coeficientes das restrições e da F.O. Estes desvios serviram para elaborar as funções de pertença, de acordo com o respectivo sinal da (in)equação e utilizando as funções triangulares descritas na secção 2.1.

A seguir, nos pontos 5.1.1 e 5.1.2, mostram-se alguns gráficos, para uma melhor visualização e análise dos resultados obtidos, anteriormente apresentados.

5.1.1 Resultados do valor da F.O versus o nível de satisfação global das condições do problema

A Figura 5.2 apresenta os resultados obtidos para os problemas das classes A e B, para um *threshold* de 30%.

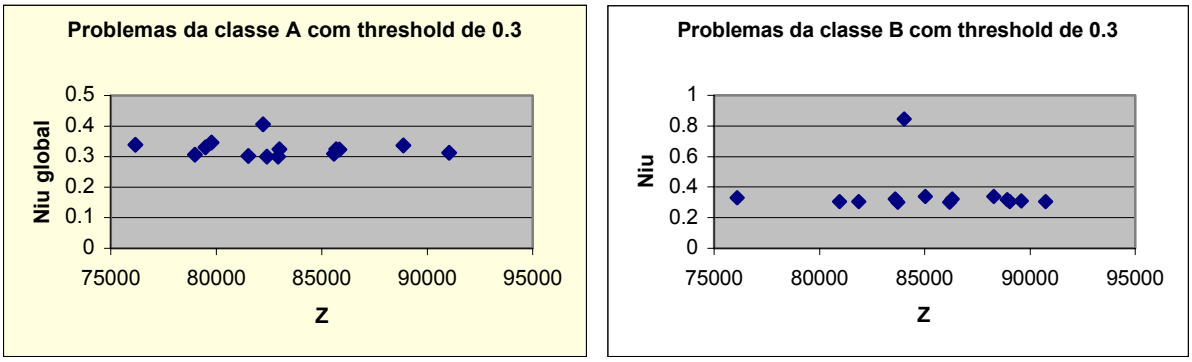


Figura 5.2 – Valores de Z versus *Niu* global para um *threshold* de 30%.

Na classe de problemas A, para um *threshold* de 30%, o problema que corresponde a um melhor valor da F.O. é o "Caso7.1A" (Z=91032.668), para um nível de satisfação dos coeficientes da F.O. e das restrições e do valor das restrições de 31.27% (*Niu*=0.3127). Na classe de problemas B, para um mesmo *threshold*, o melhor Z (90734.469) verificou-se para um *Niu* de 30.36%, no "Caso7.2B".

A Figura 5.3 apresenta os resultados obtidos para os problemas das classes A e B, para um *threshold* de 60%.

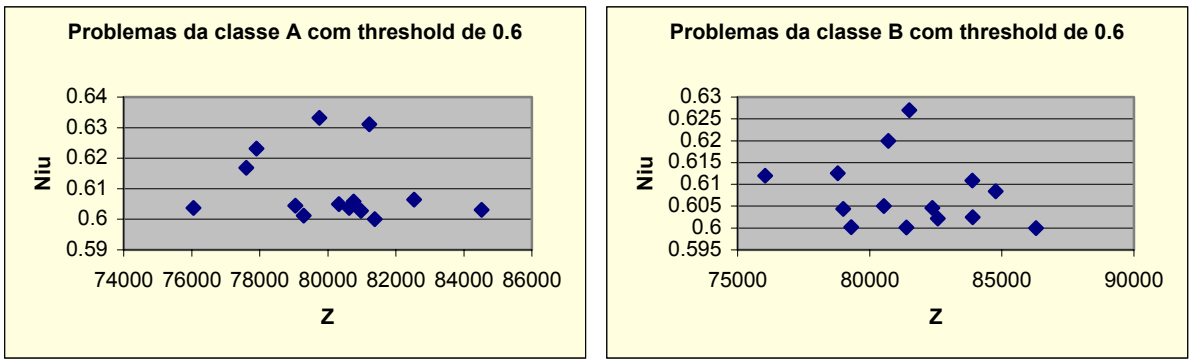


Figura 5.3 – Valores de Z versus *Niu* global para um *threshold* de 60%.

No conjunto dos problemas com *threshold* de 60%, a melhor ocorrência, da classe A, foi também o "Caso7.1A" (Z=84524.696) e da classe B o "Caso3.1B" (Z=86290.221).

A Figura 5.4 apresenta os resultados obtidos para os problemas das classes A e B, para um *threshold* de 90%.

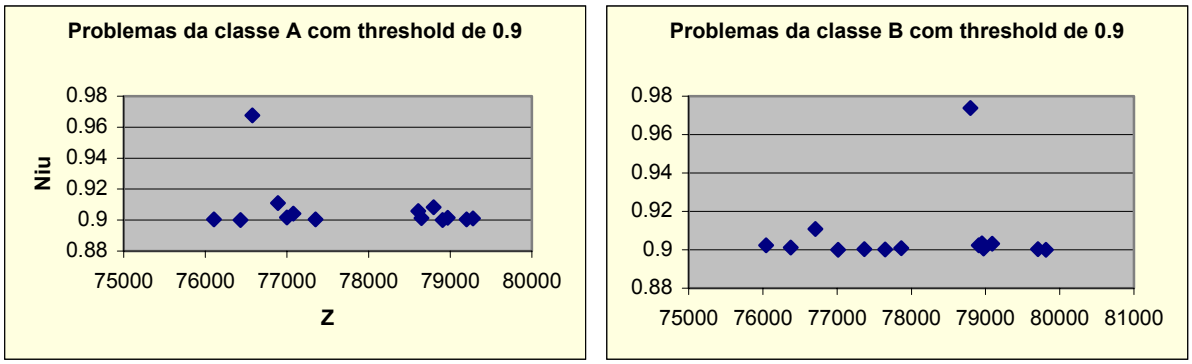


Figura 5.4 – Valores de Z versus *Niu* global para um *threshold* de 90%.

Relativamente aos problemas com *threshold* de 90% a melhor ocorrência, em termos de valor da F.O., verificou-se para o "Caso3.1A", na classe A e para o "Caso7.1B", na classe B.

5.1.2 Resultados do valor da F.O versus tempo de resolução

A seguir apresenta-se uma outra análise aos tipos de problemas considerados, desta vez com o objectivo de relacionar a variável Z (valor da F.O.) com o tempo de resposta do algoritmo a esses problemas.

A Figura 5.5 apresenta os resultados obtidos para os problemas das classes A e B, para um *threshold* de 30%.

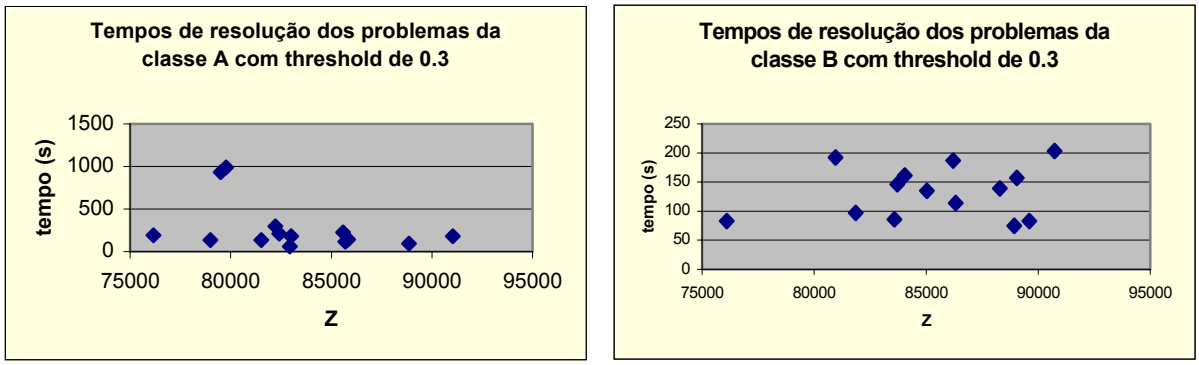


Figura 5.5 – Valores de Z versus tempos de resolução para um *threshold* de 30%.

Para um *threshold* de 30% o melhor tempo de resposta, para a classe de problemas A verificou-se para o "Caso2.1A", que foi de 59 segundos, ao qual corresponde um Z de 82935.474. Na classe B o melhor tempo registado corresponde ao problema "Caso5.1B" (Z=88934.092) e foi de 75 segundos.

A Figura 5.6 apresenta os resultados obtidos para os problemas das classes A e B, para um *threshold* de 60%.

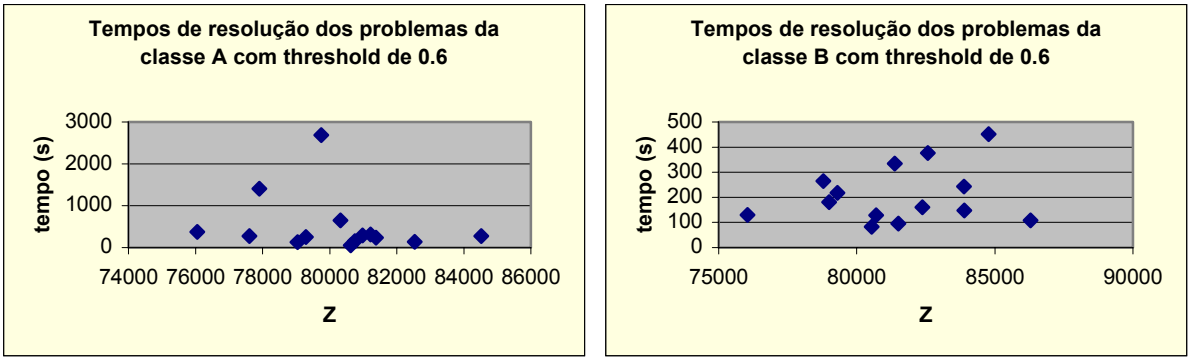


Figura 5.6 – Valores de Z versus tempos de resolução para um *threshold* de 60%.

Relativamente aos problemas com *threshold* de 60% o melhor tempo foi de 51 segundos, que ocorreu para o "Caso2.1A", nos problemas da classe A e de 83 segundos para o "Caso4.1B", para a classe B.

A Figura 5.7 apresenta os resultados obtidos para os problemas das classes A e B, para um *threshold* de 90%.

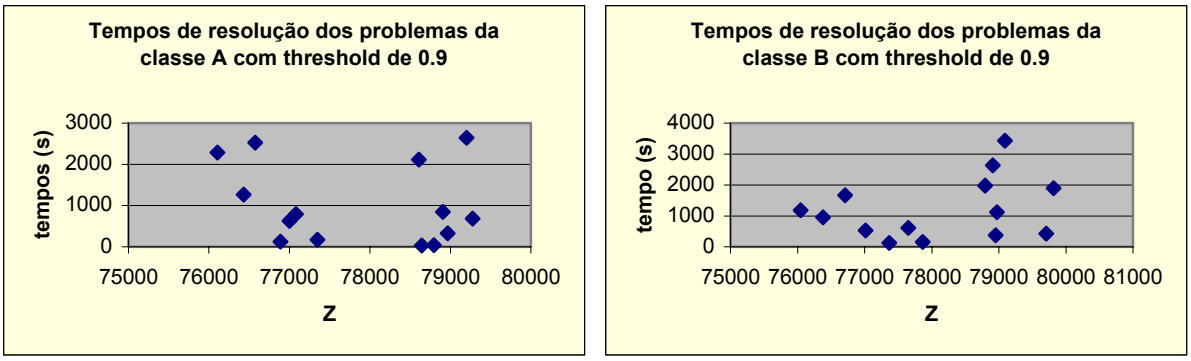


Figura 5.7 – Valores de Z versus tempos de resolução para um *threshold* de 90%.

Por último, no que se refere aos problemas com *threshold* de 90% o melhor tempo ocorreu para o "Caso4.1A", para a classe A, no valor de 33 segundos (Z=78645.183) e para a classe B foi de 124 segundos, correspondente ao "Caso2.2B", com um Z de 77367.162.

Fazendo uma análise geral final aos resultados obtidos pode constatar-se que, a um melhor valor para Z, isto é, valor da função objectivo, está geralmente associado um pior valor em termos de satisfação global das condições do problema *crisp* correspondente (*Niu* global). Também se pode verificar que todos os problemas, excepto o "Caso4.2A" e o "Caso4.2B", permitem uma satisfação plena do valor da F.O. (meta), o que indica que o valor da F.O. satisfaz a correspondente restrição. Além disso, ainda se pode concluir que o melhor resultado, em termos de valor de Z, nestas experiências, foi obtido para o "Caso7.1A", correspondente a uma "fuzificação" total (dos coeficientes e do valor da F.O. e dos coeficientes e do valor das restrições), obtendo-se um valor de 91032.668 para Z, associado a um valor de satisfação global de 0.3127 (muito próximo do limite mínimo aceitável, isto é, do valor limite imposto pelo *threshold* de 0.3, neste caso), portanto muito próximo da máxima violação total permitida nesta classe de problemas. Em contrapartida, o pior valor, em termos de Z, foi de 76104.068, para uma flexibilização do valor da F.O. e dos coeficientes das restrições (ligeiramente inferior ao correspondente valor do problema *crisp*, que é de 76250).

Poderá então concluir-se que, a melhor decisão a tomar, em cada cenário alternativo, perante os resultados obtidos, será uma decisão "flexível", na medida em que existem diferentes alternativas de decisão e nenhuma domina completamente outra. Sendo assim, a decisão final dependerá sempre de um compromisso entre tentar violar o menos possível, ou não violar nada, como se verifica para as soluções *crisp* ("*Crisp* 1" e "*Crisp* 2") ou violar o mais admissível, de modo a obter o "melhor" resultado desejado, em termos de valor da F.O., ou então optar por uma situação intermédia, em que se tenta chegar a uma situação de compromisso entre o valor obtido para a F.O. (Z) e o nível de satisfação global das "restrições" do problema (*Niu*). Como explicado anteriormente, o *Niu* é um valor agregado e consequentemente representa a melhor satisfação obtida para as restrições para o respectivo valor da F.O.

No que se refere à eficiência ou tempo de resposta do algoritmo *SA* implementado aos problemas poderá dizer-se que este apresenta comportamentos bastante diversos, ou seja, existem problemas para os quais ele consegue fornecer rapidamente uma resposta (apenas algumas dezenas de segundos), mesmo para problemas relativamente complexos, nomeadamente para os problemas pertencentes às classes de "fuzificação" incluídas no Caso 3 ("fuzificar" os coeficientes e o valor da F.O. e o valor das restrições) e no Caso 5 ("fuzificar" os coeficientes e o valor da F.O. e os coeficientes das restrições). Contudo, outras vezes verifica-se um tempo de resposta maior, que pode atingir algumas (poucas) dezenas de minutos, por vezes mesmo para problemas não muito complexos, nomeadamente, problemas pertencentes ao caso de "fuzificação" dos coeficientes e do valor da F.O. (Caso 1). Tal situação poderá justificar-se pela ocorrência de uma situação de convergência lenta, devido à entrada numa determinada zona do espaço de pesquisa, não muito favorável, para um determinado problema, que conduz a um processo moroso de resolução deste, principalmente na fase final, aquando do refinamento da qualidade da solução. Sendo assim, constatou-se que, a resolução de problemas mais complexos, isto é, que envolvem uma maior variedade de formas de "fuzificação" (e consequentemente uma maior quantidade de condições a testar) e/ou problemas de maiores dimensões (nomeadamente, um maior número de restrições – problemas que derivam do problema de base "*Crisp* 2") e/ou problemas em que o grau de exigência relativamente ao nível de satisfação das condições deste é maior (por exemplo, os casos em que o *threshold* é de 90%) não requerem, forçosamente, um maior tempo de resolução. Por exemplo, o "Caso7.2B", para um *threshold* de 90% requereu um tempo de 615 segundos (10.25 minutos), muito inferior ao tempo requerido, por exemplo, para a resolução do "Caso1.1A", para um mesmo valor de *threshold*, que foi de 2116 segundos (aproximadamente 35.27 minutos).

Em termos deste factor de avaliação do desempenho do algoritmo *SA* implementado constatou-se que, para este conjunto de problemas testado, o melhor tempo que se obteve foi de 33 segundos, que se registou para o "Caso4.1A", a que correspondeu um valor de Z de 78645.183 (que representa uma melhoria de cerca de 0.5050% em relação ao correspondente problema *crisp* – "*Crisp1"*) e um *Niu* de 90.13%, relativamente à "fuzificação" do valor da F.O. e dos coeficientes das restrições. Em contrapartida, o pior tempo verificou-se para o "Caso5.1B", no valor de 3435 segundos (57.25 minutos), para um valor da F.O. de 79091.023 (uma melhoria na ordem de 1.075%) e um nível de satisfação global das condições do problema de 90.31%, relativo à "fuzificação" dos coeficientes da F.O. e das restrições e ao valor da F.O..

6 Conclusões

Este trabalho mostra que o algoritmo *SA* é uma boa técnica de resolução para tratar problemas de optimização difusos, podendo as suas vantagens sumariar-se do seguinte modo: simplicidade e fácil implementação; independência em relação ao problema a tratar (pode resolver problemas lineares e não lineares); não requer transformações matemáticas do problema a tratar para o resolver (como é o caso do método de Zimmerman). Note-se que na implementação do algoritmo *SA* foi, adicionalmente, introduzida a possibilidade, por parte do utilizador, de escolher uma semente para a geração dos valores aleatórios, através de um processo de selecção pseudo aleatório desses valores, subjacente a tal geração. A principal desvantagem do algoritmo *SA* é a necessidade de definir os estados iniciais que satisfazem as restrições, para

cada variável. Outra desvantagem é a escolha de uma temperatura (T) adequada, porque pode implicar uma pesquisa maior e consequentemente mais tempo de computação.

Além dos parâmetros Z (valor da F.O.) e *Niu* (nível de satisfação global das condições do problema) achou-se também conveniente avaliar o tempo de resposta do algoritmo aos problemas, sendo este um *output* adicional fornecido pela aplicação desenvolvida. Relativamente a este aspecto será conveniente referir que o tempo de resposta aos problemas não apresentou um comportamento linear e claramente previsível, mas antes um comportamento que dependia, fortemente, do tipo de convergência verificado para cada problema. Sendo assim, verificaram-se situações em que, apesar de uma relativa complexidade do problema tratado o algoritmo fornecia uma resposta rápida ao problema (ou mesmo quase imediata) e vice versa. Daqui se pode concluir que o tempo de resposta do algoritmo *SA* embora também dependa, em geral, do tipo de problema considerado (por exemplo, do caso de "fuzificação" e da dimensão do problema) depende essencialmente do tipo de convergência verificado, isto é, da zona do espaço de pesquisa para onde um determinado problema é canalizado, em cada execução do algoritmo (o que também é influenciado pela escolha da semente) e do nível de satisfação das condições do problema (*threshold*). Sendo assim, enquanto que o primeiro factor é, de certa forma, mais imprevisível e, desta forma, conduz a um problema mais dificilmente contornável, o segundo factor apresenta um crescimento na razão directa do tempo de resposta do algoritmo, ou seja, o tempo de resposta aumenta com o aumento do valor de *threshold*, dentro de cada classe de problemas, como também seria de esperar, pois aumenta o nível de exigência dos resultados obtidos, em termos de satisfação das condições do problema.

O tipo de convergência verificado pelo algoritmo implementado e o correspondente desempenho, em termos de tempo de resposta aos problemas é, de certa forma, um aspecto característico dos métodos de pesquisa não determinísticos, como é o caso do método de *Simulated Annealing*. Uma forma de tentar melhorar esta situação será através da introdução de determinadas heurísticas, adequadas a um determinado tipo de problemas, enquadrados num determinado contexto específico, o que conduzirá a um processo de adaptação mais rigorosa do algoritmo às suas necessidades de utilização concretas, ditadas numa determinada situação particular.

O método de "fuzificação" proposto por Ribeiro & Moura-Pires foi implementado com o algoritmo de resolução *SA* e ilustrado com um conjunto de exemplos, que mostram quão flexível e adaptável o método é na resolução de problemas de optimização difusos. Como se mostrou, a principal vantagem do uso deste método consiste na liberdade de que o decisor dispõem para escolher qualquer modelo que este considere apropriado para um problema específico.

A fim de permitir tratar também problemas do tipo inteiro, tornou-se necessário incluir a opção de proceder a um arredondamento dos valores (componentes) do vector de variáveis de decisão geradas.

Através da análise de resultados efectuada verificou-se que os melhores resultados, em termos de valor da F.O. foram de Z=91032.668, para a classe de problemas A e Z=90734.469, para a classe B e foram obtidos com a "fuzificação" total (Caso 7), isto é, "fuzificação" dos coeficientes e valor da F.O. e dos coeficientes e do valor das restrições, mas à custa de uma violação ou insatisfação das condições (*threshold* de 30%). Deve também notar-se que, em geral, se o decisor pretender atingir um melhor nível de satisfação das condições, isto é, dos parâmetros ou valores das restrições e/ou do valor da F.O. e/ou da satisfação dos valores dos coeficientes da função objectivo e/ou das restrições, isso irá conduzir a um pior resultado em termos do resultado esperado para a função objectivo e vice versa.

Em termos de trabalho futuro seria interessante extender este estudo, através de uma comparação dos resultados aqui obtidos com os resultados para outros valores para os parâmetros de controlo do algoritmo, nomeadamente a temperatura inicial e eventualmente tentar também ajustar melhor a forma de variação (diminuição) da temperatura, face à natureza dos problemas específicos que se pretendem resolver (nomeadamente, a dimensão do problema e os casos de "fuzificação" que se pretendem abordar), adicionalmente à averiguação da possibilidade de introdução de heurísticas, para facilitar o processo de resolução dos problemas a tratar.

Um outro aspecto interessante seria avaliar o comportamento do algoritmo na resolução de problemas não lineares.

Além disso, também seria interessante tentar ajustar, mais rigorosamente, este algoritmo para o caso das variáveis do tipo inteiro, através da adaptação das funções de pertença para valores discretos, de modo a tornar o algoritmo mais rigoroso e adequado à resolução deste tipo de problemas.

Referências

[1] Aarts E., Lenstra J. K., "*Local Search in Combinatorial Optimization*", John Wiley & Sons, Inc., 1997.

[2] Bellman R. E., Zadeh L. A., "*Decision-Making in a Fuzzy Environment*", Management Science, Vol. 17, N.4, Dezembro, 1970.

[3] Blazewics J, Ecker K.H., Pesch E., Schmidt G., Weglarz J., "*Scheduling Computer and Manufacturing Processes*", Springer-Verlag, 1996.

[4] Bundy A., "*Artificial Intelligence Techniques*", Springer-Verlag, 1997.

[5] Lai Y. J., Hwang C. L., "Fuzzy Mathematical Programming - Methods and Applications", Springer-Verlag, 1992.

[6] Lai Y. J., Hwang C. L., "Fuzzy Multiple Objective Decision Making - Methods and Applications", Springer-Verlag, 1994.

[7] Moura-Pires F., Ribeiro R. A., "A New Risk Function for Fuzzy Linear Programming", Proceedings of the World Automation Congress (WAC'98), N.138, Alaska, Maio, 1998, 1-10.

[8] Ribeiro R. A., Pires F. M., Pires J.M., "Solving Fuzzy Optimisation Problems: Flexible Approaches using Simulated Annealing", Proceedings of the World Automation Congress (WAC'96), Montpellier, France, Maio, 1996.

[9] Ribeiro R. A., Pires F. M., "Fuzzy Linear Programming via Simulated Annealing", *Kybernetica*, Vol. 35, N.1, 1999, 57-67.

[10] Zadeh, L. A., "Fuzzy Sets" *Information and Control*, N.8, 1965, 338-353.

[11] Zimmermann H. J., "*Fuzzy Set Theory and its applications*", Kluwer Academic Publishers, 1991.