Proceedings of the 10th Mediterranean Conference
on Control and Automation - MED2002
Lisbon, Portugal, July 9-12, 2002.

SIMULATED ANNEALING & FUZZY OPTIMIZATION

L.R. Varela[*], R.A. Ribeiro[†], and F.M. Pires[+]

[*]Leonilde Rocha Varela, University of Minho, School of Engineering, Dept. of Production & Systems,
4800-058 Guimarães, Portugal
E-mail: leonilde@dps.uminho.pt

[†]Rita Almeida Ribeiro, Lusiada University, Dept. of Econo. & Man. Science
1349-001 Lisbon, Portugal
E-mail: rar@lis.ulusiada.pt

[+]Fernando Moura Pires, University of Évora, Dept. of Computer Science
7000 Évora, Portugal
E-mail: fmp@uninova.pt

Keywords: Fuzzy Linear Programming, Fuzzy Optimization, Simulated Annealing.

Abstract

The Simulated Annealing (SA) algorithm is an adequate technique for solving fuzzy optimization problems, through the selection of the best solution among a finite number of possible solutions. Here, we present the results for a set of fuzzy problems, selected with the purpose of testing the SA algorithm performance, which has been implemented in this work. 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 of the SA algorithm, for the resolution of fuzzy linear optimization problems.

1 Introduction

The Simulated Annealing (SA) algorithm is a particularly attractive technique to solve fuzzy optimization problems, because it allows finding close to optimal solutions, which, without big computational effort, in a fuzzy environment, is usually good enough. In this work we have chosen a set of problems in order to test de performance of a SA algorithm, which has been implemented. The problems tested were formulated following a complete fuzzification method proposed by Ribeiro and Moura-Pires (1999) [4].

In this work, the main objective consists on showing the suitability of the implemented SA, for the resolution of fuzzy linear optimization problems. The principal evaluation measure of its performance consists on the value obtained for the objective function, in the presence of a certain trade-off situation and considering certain satisfaction level for the constraints that characterize the problem (the threshold level). Other evaluation measure is the time to achieve a solution. Further, the parameters that control the SA algorithm are also discussed to show how easily they can be manipulated.

Another objective is to illustrate the flexibility of the optimization method proposed by Ribeiro and Moura-Pires in the formulation of several types of fuzzy linear optimization problems with single or multiple objectives.

The structure of the remaining sections of this paper is the following. In section 2 we present a brief introduction to fuzzy optimization. In section 3 the SA algorithm is described and we present an explanation of the algorithm's implementation details. In section 4 we present the set of cases tested that will be analyzed (covering a total of 42 fuzzy problem cases). Section 5 presents the results for the cases that were considered and solved with the implemented SA algorithm, as well as a

brief trade-off analysis. Finally, section 6 presents some conclusions and future work is outlined.

2 Fuzzy optimization

In general we can say that Fuzzy Decision Making integrates two fundamental topics, which are the "Fuzzy Multi-attribute" and "Fuzzy Optimization" [5,6]. In this paper we focus on Fuzzy Optimization, specifically when the coefficients and/or the resources and/or goals are imprecise/fuzzy [4].

Fuzzy Set Theory was first developed by Zadeh [9]. Bundy [1] defines Fuzzy Set Theory as an extension to the conventional Set Theory in which the membership degree for an element in a set takes a value in the interval [0,1], instead of just 0 or 1. A Fuzzy set is a set (class) of objects in which there isn't defined a clear "border" between the objects that belong to the class and the ones that do not belong to it. A more precise definition can be formulated as follows [12]:

Let X={x} a class of objects (points) generically represented by x. Then a fuzzy set A in X is a set of ordered pairs, A = {(x, μA(x))}, x∈X, where μA(x) is designated by the membership degree of x in A, and x is an element of X. Generally we assume that μA(x) belongs to the interval [0,1], where 0 and 1 represent, respectively, the membership levels, or degrees, of Boolean Logic. So, we assume that a fuzzy set can be defined accurately, by associating to each object x a value between 0 and 1, which represents its membership degree in the subset A.

Classical optimization problems consist on maximizing or minimizing a certain objective function subject to a set of constraints, which express, for example, the resources' limitation. Formally, max f(x) subject to x∈X.

The first model to solve decision problems in a fuzzy environment was proposed by Bellman and Zadeh [12]. These authors consider that the decision on each alternative results from the intersection of the objectives and the constraints and that the union of the decisions on each alternative gives the "best" decision. A fuzzy decision can then be seen as an intersection of goals and constraints, which means that it is a symmetric model. A maximization fuzzy decision is defined as a point in the space of alternatives, in which the membership function of a fuzzy decision reaches its maximum value [12].

The main objective of fuzzy optimization consists on finding the "best" solution (decision alternative) in the presence of incomplete information, i.e., imprecise information and/or in the presence of "vague" limits in the information. There exist many forms of imprecision in fuzzy optimization problems, as for example, variable coefficients that we don't know precisely (for example, "processing times of about one hour, for assembling a piece") and constraint satisfaction levels with imprecise limits (for example, "the total processing time available is around 100 hours"). At present the challenge is on allowing the construction of models that allow interpreting vague and imprecise concepts, which can be translated in fuzzy quantitative methods [11].

A classical linear optimization problem can be formalized as,

$$\max/\min \quad Z = Cx \;,\; Ax \le B,\; x \ge 0 \qquad (1)$$

The fuzzy version of this problem is generally formalized as,

$$\max/\min \quad \tilde{Z} = \tilde{C}\,x \;,\; \tilde{A}\,x \,\{\ge, \le, =\}\, \tilde{B} \;,\; x \ge 0 \qquad (2)$$

where $\tilde{Z}$ represents a fuzzy goal, $\tilde{C}$ is the vector of fuzzy costs, $\tilde{A}$ is the matrix that contains the fuzzy coefficients of the objective(s) and of the constraints and $\tilde{B}$ is the corresponding vector of the limits/bounds of the objective(s) and of the resources. The "tilde" on top of the parameters means that they are defined as fuzzy sets. Alternatively we can use the tilde on top of the equation signal $\{\tilde{\ge}, \tilde{\le}, \tilde{=}\}$.

In general, the process of solving a fuzzy optimization problem consists on the following steps:

- Depending on the problem type, formalize it in the form of a linear programming problem or in the form of a multi-objective problem, or even as a non-linear programming problem.
- If we intend to fuzzify the objectives, define the goal for those objectives.
- Select the membership functions to represent the fuzzification of any parameter and constraints, such as triangular, sinusoidal, trapezoidal, or other.
- Define the membership functions with the necessary parameters for the preference value and tolerances.
- Define thresholds for the allowable degree of the deviations/violations in the constraints satisfaction.
- Define the aggregation operator to combine the constraints (and goals), e.g. the t-norm operator minimum.
- Solve the problem (in this work with a SA algorithm).

The first proposal to formalize a fuzzy linear programming problem is due to Zimmermann in 1976 [6]. This author considers that there can exist flexibility in the violation of both constraints and objective(s). This model is based on the symmetric model of Bellman and

Zadeh (i.e., there is no difference between constraints and objectives). However, at present there exist several fuzzification methods proposed in the literature, for the resources and goals' fuzzification, as well as for the fuzzification of the coefficients (see details in [13]).

In this paper we are only interested in testing the case of complete fuzzification because we want to show the flexibility of the simulated annealing to solve linear or non-linear problems in a fuzzy environment.

2.1 Complete flexibilization model

Several authors proposed different formalizations to solve the case of complete fuzzification, as for example Carlsson and Korhonen (mentioned in [13]). In this work we only discuss in detail the method proposed by Ribeiro & Moura-Pires [4,5,11], because it is a flexible approach that can deal with single or multiple objectives, as well as linear or non-linear programming problems.

Ribeiro and Moura-Pires [4,5,11] proposed a method for fuzzy optimization problems that include fuzzy coefficients either in the objective function or in the constraints. The whole process is designated as "flexible approach" for fuzzy optimization problems and its formalization is: assuming the general formulation for fuzzy optimization problems defined in (2),

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

this is transformed into the following system of non-linear equations:

$$\begin{cases} \max\limits_{x} \quad Z = \sum\limits_{k=1}^{K} w_k \cdot x_k \\ \begin{cases} \max\limits_{y,w} \quad M = \cap(\mu_{aik}, \mu_{bi}, \mu_{ck}) \\ \sum\limits_{k=1}^{K} y_{i,k} \cdot x_k \{=, \leq, \geq\} \tilde{b}_i \\ y_{i,k} = \tilde{a}_{i,k} \\ w_k = \tilde{c}_k \end{cases} \end{cases} \tag{3}$$
$$k = 1,..., K \text{ e } i = 1,..., N \text{ e } x, y, w \geq 0$$

It should be noted that all the fuzzy parameters are presented with a tilde on the top. The $\cap$ symbol represents the intersection of all membership values. The second objective represents the maximization of the intersection of all membership values, i.e. this goal gives the best value of the minimum (intersection) of the violations of constraints.

In this flexibilization process the fuzzy values c_k and a_{ik} become process variables. It can even happen that one of these coefficients is dependent on another c_k or a_{ik}. These dependences can easily be incorporated in the model by expressing the existent relationships. Notice that with this new method only the independent variables are variables of the optimization problem.

The membership functions used in this paper are triangular functions. With this type of functions we can easily represent any fuzzified parameter of the optimization problem. Considering a deviation p_i for the flexibilization, the three types of functions that are more commonly used to represent less or equal, greater or equal, or just equal constraints (signals), are:

$$\mu_i(x) = \begin{cases} 0 & if \quad A_i(x) > b_i + p_i \\ 1 - \dfrac{A_i(x) - b_i}{p_i} & if \quad b_i < A_i(x) \leq b_i + p_i \\ 1 & if \quad A_i(x) \leq b_i \end{cases} \tag{4}$$

$$\mu_i(x) = \begin{cases} 0 & if \quad A_i(x) < b_i - p_i \\ 1 + \dfrac{A_i(x) - b_i}{p_i} & if \quad b_i - p_i \leq A_i(x) < b_i \\ 1 & if \quad A_i(x) \geq b_i \end{cases} \tag{5}$$

$$\mu_i(x) \begin{cases} 0 & if \quad A_i(x) < b_i - p_i \\ 1 + \dfrac{A_i(x) - b_i}{p_i} & if \quad b_i - p_i < A_i(x) \leq b_i \\ 1 - \dfrac{A_i(x) - b_i}{p_i} & if \quad b_i < A_i(x) < b_i + p_i \\ 0 & if \quad A_i(x) \geq b_i + p_i \end{cases} \tag{6}$$

Function (4) corresponds to the membership function for constraints «less or equal»; function (5) is the membership function for constraints of the «greater or equal» type; and function (6) is the membership function for constraints of the «equal» type.

Function (6) can also be used to represent fuzzy coefficients. Note that in the Zimmerman's method [6] only the functions of greater or equal than or of less or equal than, respectively Figures 1 and 2, are used to define or represent either the goals of the objective or the constraints. It is through these functions that the mathematical transformation proposed by Zimmerman [6] is possible and maintains the linearity in the transformed problem. When we use algorithms which are independent of the problem for its solution, namely the SA algorithm, any other function can be used, as for

example a sinusoidal or a trapezoidal function, because they do not require linearity of problems [11].

The solution method (3) proposed in this approach relies on two fundamental steps:

1st) To find the best values for x, y and w, independent variables that maximize the minimum aggregated (M) value, considering a threshold value, pre-defined by the decision maker, as the minimum acceptable violation of the constraints.

2nd) To find the optimal value of Z that satisfies all the constraints as well as the first step.

In conclusion we can say that this fuzzification model provides trade-offs between constraints' satisfaction and the value of the goal to reach. Further, this method allows manipulating any type of linear or non-linear fuzzy optimization problem. Another advantage of this method is the possibility to solve crisp problems that have no solution by including imprecision on the coefficients or on the parameters.

3 Optimization resolution algorithms

Among the enormous variety of optimization algorithms it is possible to identify a set of procedures that can be grouped in three fundamental categories [7]:

- Exact or approximate mathematical procedures;
- Simulation-based procedures;
- Heuristic procedures.

The exact mathematical procedures correspond to exact solution algorithms that have the objective of finding the optimal solution. Three fundamental classes of these algorithms are the exact Linear Programming Algorithms, Dynamic Programming Algorithms and Branch and Bound (B&B) Algorithms. These algorithms, being exhaustive, aren't appropriate for solving problems of great dimensions or difficult problems. Furthermore, many optimization problems are NP-complete. Therefore, there exist heuristic approximate algorithms that are more appropriate for the solution of "difficult" problems. Although heuristic methods don't guarantee optimal solutions, they are usually faster and they allow obtaining good solutions or at least acceptable solutions in a reasonable or acceptable amount of time, and this idea has motivated the accomplishment of this study. Most of these algorithms are relatively recent and have been increasingly explored. The more known heuristic methods are Simulated Annealing, Taboo Search as well as Genetic Algorithms.

This work is centred in one of the algorithms belonging to the class of local search algorithms or guided random search techniques, more precisely, on the SA technique.

3.1 SA algorithm

The SA is a search technique that belongs to the general class of guided random search techniques [8], which are based on enumerative techniques but use additional information to guide the search. Guided random search techniques are quite general in scope and can solve complex problems. This type of algorithms play an important role in the scope of local search techniques, for two major reasons [3]: first, because they have been proved successful when applied to a large number of practical problems; second, some of those algorithms, such as the SA algorithms, have a stochastic component, which facilitates a theoretical analysis of its asymptotic convergence and this made them quite popular among mathematicians.

The objective of an algorithm of this nature is to find the best solution among a finite number of possible solutions. The SA technique is particularly attractive, because it allows finding near-optimal solutions with reasonable computational effort. Notice that in this algorithm it is not possible to know if the best solution found is indeed the global optimum. This characteristic restricts its use to the cases where good local optima are acceptable [7].

In SA procedures, the sequences of solutions don't tend linearly to local optima, as it happens in other local search techniques. Conversely, we can verify that the solutions trace a variable course, through a set S, of possible solutions and this course tends to be guided in a "favourable" direction.

In general, we can say that the SA algorithm is a good procedure to use in situations in which the knowledge is imprecise. Even for complex problems this technique is relatively easy to implement and it usually executes a hill-climbing procedure, with multiple restarts. Typically, this technique generates a "Markovian" network in which the successor of a current point is stochastically chosen, with a probability that only depends on the current point and not on the previous history of the search.

As mentioned previously, the SA algorithm is an algorithm used for the solution of optimization problems, where the objective function (O.F.) corresponds to the energy of the states of a solid. The SA algorithm requires the definition of a neighbourhood structure, as well as the parameters for the cooling programming. A temperature parameter allows distinguishing among deep or slight alterations in the O.F. Drastic alterations occur at high

temperatures and small or slight modifications at low temperatures. It is an "evolutionary" process, that moves in small steps, from a state to another, and there exists the problem of being "arrested" by a local optimum [11].

When we use the SA algorithm for solving fuzzy optimization problems, the decision maker can select the thresholds levels (α-cuts) as well as the tolerances (deviations) for each constraint parameter and/or for each objective function's coefficient and/or for each constraint's coefficient. The membership functions are then built with the tolerances defined. Furthermore, the SA algorithm allows dealing with any type of fuzzification.

3.2 SA implementation

According to Ribeiro and Moura-Pires [11], the four basic requirements for using the SA algorithm in combinatorial optimization problems are:

- A concise problem description;

- A random generation of the alterations from one configuration to another;

- An objective function that includes the utility function of the trade-offs;

- A definition of the initial state, of the number of iterations to be executed for each temperature and its annealing process.

The aggregation function (M) is the intersection of all the membership values of the constraints and the operator used for the intersection was the "min"(see (3)). The intersection represents the logical "and", to indicate that all the constraints have to be satisfied with a certain level (min). In addition, a threshold value was included to allow the decision maker to specify if he wants the constraints to be satisfied with "at least" a certain value. This threshold corresponds to the α-cut mentioned in section 3.1. Summarizing, the objective to attain for the M is to maximize the minimum of the constraints deviations (membership values) or to maximize the aggregation/ intersection of the membership degrees of the fuzzy constraints, which corresponds to the best possible value of the smallest level of constraints satisfaction (as shown in (3)).

The variable δ expresses the difference between the previous value of M (membership value of the aggregation) and the new value obtained (current M) and it is used to choose the maximum M, i.e., the best satisfaction level for the combined constraints. This variable represents the energy difference between the previous state and the new one. The probability function for moving to a better energy state (best objective) is given by the exponential of δ divided by the control parameter T. The smaller the temperature T is, the less probable will be an occurrence of some alteration. T(t) is the temperature decrease function. N(t) represents the number of neighbour solutions generated and, consequently, the number of possible solutions.

The selection of options and parameters for the SA algorithm implementation were based on the work presented in [14]. Hence, in the SA algorithm implementation we specified: how to generate a state y, neighbour of x; which aggregation function (M) to use; the number of neighbours generated, N(t); the temperature decrease function, T(t) and finally the algorithm stopping criteria.

We also included in the algorithm the notion of seed [2] for the random numbers generation, to allow the generation of identical values, in different program executions, for the same example. This seed is very useful because it allows us to compare the different results obtained, when small variations in the execution constraints of each problem occurred. For our tests we used a seed value of 1.

The choice on how to generate a state y, neighbour of x, is done through the definition of a new state, which is a random point y, in which the distance of that new point to the current point z is random and lower than ρ(t), defined by:

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

The neighbour points are generated through the use of polar coordinates that guarantee the necessary randomness of the selection process for the new points considered. For n decision variables we have for Δx,

$$\Delta x_1=\rho\times\cos\theta_1 \quad \theta_1\in[0,2\pi]$$

$$\dots$$

$$\Delta x_k=\rho\times\sin\theta_1\times\dots\times\sin\theta_{k-2}\times\sin\theta_{k-1}\times\sin\theta_k \ \theta_k\in[0,\pi]$$

The new point, in the neighbourhood of the previous point is then given by:

$$(y_1,\dots,y_n)=(x_1,\dots,x_n)+(\Delta x_1,\dots,\Delta x_n)$$

We should note that, after the generation of a new point, in the neighbourhood of the last one, it is necessary to test if that new point (vector of values, y) satisfies the constraints of the fuzzy problem.

The selection of the number of neighbours to generate N(t), follows the following heuristic:

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

This heuristic takes under consideration that more successors should be generated when the temperature T decreases, in order to have more options to test. The temperature function T(t) is a function that uses a decrease factor of 0.99. The stopping criterion of the algorithm is reached when the temperature is less or equal than 0.0001.

4 Linear optimization problems tested

In this work we tested a set of problems, which can vary in the type of fuzzification considered, in the deviations allowed in the constraints of the problem, in the threshold level, and/or in the problem dimension, among other kind of parameters that control the SA algorithm, which can also vary.

4.1 Crisp linear problems formulation

To start, two kind of crisp problems were considered, as basic problems for the formulation of fuzzy problems. The first problem crisp 1 (C1) considers 10 constraints, from constraint 11 to the constraint 20, and the second problem crisp 2 (C2) includes all the constraints (22), as depicted in Table 1.

#	LHS							RHS
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
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

Table 1: Constraints of the crisp problems.

To explain Table 1 we remark that for example constraint #5 means:

$$x_1 + 1.67x_2 + 1.43x_3 + 1.25x_4 + 1.11x_5 + 1.25x_6 + x_7 \leq 6000$$

To perform the evaluation tests we used the same objective function for both crisp problems, as follows:

$Z = \text{Max } (200\ x_1 + 300\ x_2 + 400\ x_3 + 500\ x_4 + 600\ x_5 + 600\ x_6 + 650\ x_7)$, $x_j \geq 0$ (j = 1, 2, ..., 7).

In addition, we solved the two deterministic versions of the problems with the simplex algorithm to obtain the values for the goals (which will be fuzzified in some tests). The results obtained for problem C1 and C2 are shown in Table 2:

Pr.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Z
C1	10	20	20	0	55	0	45	78250
C2	10	20	20	20	10	25	45	76250

Table 2: Results for the crisp problems.

As expected, the version with less constraints (C1) has a larger maximum value for the objective, 78250, than the version with more constraints (C2), which has 76250, even though some of the optimal x_i are the same in both solutions, presented in Table 2.

4.2 Problems fuzzification/flexibilization

As mentioned we can have several types of fuzzification in linear optimization problems. In this work we considered the seven following flexibilization cases:

Case1: Fuzzify the coefficients of the O.F. and the goal.

Case2: Fuzzify the goal and the constraints limits.

Case3: Fuzzify the O.F. coefficients, the goal and the constraints limits.

Case4: Fuzzify the goal and the constraints coefficients.

Case5: Fuzzify all the coefficients (O.F. and constraints) and the goal.

Case6: Fuzzify the goal, the constraints coefficients and the constraints limits.

Case7: Fuzzify everything (coefficients of the O.F. and constraints, goal and constraints limits).

These types of fuzzification were then applied to each crisp formulation, resulting in a total of 14 problems. To create all the membership functions we used deviations of 10%. The 14 fuzzy problems were tested under 3 different scenarios, in terms of the threshold value imposed in the satisfaction level of the constraints, corresponding to 30%, 60% and 90%. Thus, a total of 42 different cases were tested, with the purpose of obtaining

a sufficiently detailed and diversified analysis for the results obtained with the SA algorithm.

In agreement with the formalism presented in (3), when a problem includes flexible coefficients in the objective function, or in the constraints and/or we consider the fuzzification of the constraints' parameters (and/or of the O.F. value - goal) we will use a tilde, placed on the top of the correspondent coefficient and/or sign of the "equation", as for example, $\tilde{0.33}$ and $\tilde{\leq}$.

5 Fuzzy optimization results

The development of the software application enables running the SA algorithm for different threshold values and the objective consists on allowing a comparison among different results obtained, for the same problem, in order to analyze if it is possible to converge to a certain point. So, we can try to reach some conclusions, which correspond to certain trade-off situations, in terms of (3) for the O.F. value (Z) and for the global satisfaction level of the problem constraints (M).

It is also important to remember that the examples tested were executed for threshold values of 0.3, 0.6 and 0.9, for the different fuzzification cases considered. We also implemented a clock, which allows us to evaluate the time (in seconds) to reach an answer (the algorithms' solution time).

5.1 Results analysis

The following tables present the results obtained for the tested problems. They include, the value of the O.F. (Z), the corresponding problem constraint satisfaction level (M) and the time it took to achieve the solution. It is also important to mention that M represents the minimum constraints violation level, i.e., the best satisfaction level found for the O.F. value (see equation (3)). The problem designations contain information concerning the tested cases. For example: Case **1**.Y, corresponds to the type of fuzzification used (1 to 7 as described in sub-section 4.2, in this example Case 1). Case x.**1** corresponds to the problem Crisp1 (C1) or Crisp 2 (C2), in this example it corresponds to C2.

The results of the tests, shown in the next tables, correspond to the 7 fuzzification cases presented in point 4.2. The tests were performed using an initial temperature of 0.12 and we used 10% deviations (tolerances) to construct the membership functions, according to (4), (5), (6).

Threshold 0.3			
Cases	Z	M	Time
Case1.1	82223.584	0.4059	297
Case1.2	79775.478	0.3458	990
Case2.1	82935.475	0.3003	59
Case2.2	81520.987	0.3019	135
Case3.1	88864.288	0.3366	93
Case3.2	85833.658	0.3237	142
Case4.1	79496.303	0.3310	934
Case4.2*	76163.606*	0.3385	193
Case5.1	85689.700	0.3238	116
Case5.2	78994.391	0.3065	137
Case6.1	82995.997	0.3240	182
Case6.2	82411.910	0.3001	210
Case7.1	91032.668	0.3127	179
Case7.2	85582.212	0.3100	226

Table 3: Results for threshold of 0.3.

Threshold 0.6			
Cases	Z	M	Time
Case1.1	80321.289	0.6050	648
Case1.2	77901.232	0.6232	1402
Case2.1	80623.626	0.6037	51
Case2.2	79040.269	0.6044	129
Case3.1	82535.396	0.6064	140
Case3.2	80753.384	0.6058	154
Case4.1	79748.248	0.6332	2686
Case4.2*	76055.217*	0.6037	372
Case5.1	81218.748	0.6311	314
Case5.2	77604.922	0.6169	273
Case6.1	81379.872	0.6000	240
Case6.2	79290.528	0.6012	255
Case7.1	84524.696	0.6031	277
Case7.2	80973.784	0.6028	293

Table 4: Results for threshold of 0.6.

Threshold 0.9			
Cases	Z	M	Time
Case1.1	78606.265	0.9059	2116
Case1.2	76575.161	0.9676	2524

Case2.1	78796.001	0.9083	42
Case2.2	76889.279	0.9109	126
Case3.1	79276.940	0.9010	681
Case3.2	77348.281	0.9005	169
Case4.1	78645.183	0.9013	33
Case4.2*	76104.835*	0.9005	2287
Case5.1	78967.500	0.9014	329
Case5.2	76433.001	0.9001	1267
Case6.1	78907.454	0.9001	850
Case6.2	77000.635	0.9016	631
Case7.1	79200.534	0.9004	2643
Case7.2	77079.792	0.9041	796

Table 5: Results for threshold of 0.9.

As can be observed, in Tables 3, 4 and 5, the problem Case4.2, marked with '*', correspond to the only example, among the ones analyzed, in which there were no improvements in the O.F. value (Z) when compared with the corresponding crisp solution. The values highlighted correspond to the maximum values for the O.F.

Next we will proceed with a detailed analysis of the results illustrated in the previous Tables. So we will explain the trade-offs between the value of the O.F. versus the satisfaction of the constraints and coefficients and also describe the trade-offs between the value of the O.F. and the time to achieve the result.

O.F. value vs. constraints global satisfaction level

For a threshold of 30%, the problem with the "best" O.F. value was Case7.1 (Z=91032.668), for a satisfaction level of coefficients and constraints of 31.27% (M=0.3127).

In the set of the problems with threshold of 60%, the best occurrence was also Case7.1 (Z=84524.696). We can observe that the value of the objective function becomes worse for a better satisfaction level for constraints and coefficients (M). The trade-off is quite high and it shows clearly that if the decision maker wishes to have better satisfaction values for the M he will, generally, have to "pay" in terms of worse objective value, Z.

Relatively to the problems with threshold of 90% the best result in terms of the O.F. value, was obtained for Case3.1. Observing Table 5 it is clear that there is only one solution that has very good values for satisfaction of constraints and coefficients but at the expense of lower values for the O.F. The type of question the decision maker will be faced with is: is it better to have a Z= 76,575.161 with a M of 0.97 (crisp C2) or is it better to have a Z= 79,276.940 with a decrease in satisfaction value to M = 0.90 (crisp C1)?

We can also verify that all the problems, except Case4.2, yield higher values for the O.F. then the values obtained with the crisp version. This shows that we can obtain substantial gains by flexibilizing some optimization problems. Furthermore, the best result obtained in terms of Z was for Case7.1 (91032.668), which corresponds to the total fuzzification of the problem (O.F. value, coefficients and constraints). This Z value is associated with a global satisfaction level of 0.3127, which is close to the maximum total violation allowed, which was 0.3 (threshold limit). The conclusion is, the more we flexibly, the more benefits we can expect. However, we will have to pay, either in terms of time or satisfaction level of constraints or coefficients to achieve these gains. Another interesting result (Case 4.2) is that the worst value in terms of Z (76104.068) corresponds to the flexibilization of the O.F. value and the constraints coefficients; hence, for the problems we tested there is no advantage of flexibilizing just these parameters.

O.F. value vs. solution time

To test the computational time of the SA algorithm we used a personal computer with a PentiumIII processor, working at a velocity of 450 MHz.

In terms of the SA evaluation factor, computational time, it was verified that for the group of problems tested, the fastest time obtained was 33 seconds, which occurred for Case4.1, to which corresponded a Z of 78645.183 (which represents an improvement of about 0.5050% in comparison to the corresponding crisp problem, "C1") and a M= 90.13%, relatively to the fuzzification of the O.F. value and of the constraints coefficients. On the other hand, the largest time occurred for Case4.1, with a value of 2686 seconds (44.7 minutes), for a Z of 79748.248 and a global satisfaction level of 63.32%.

Regarding the efficiency of the implemented SA algorithm solution time, some problems reached a solution quickly (just some dozens of seconds), even for relatively complex problems, namely the problems belonging to Case 2. This seems to show that when there is no need to fuzzify the coefficients of the independent variables of the constraints the results are obtained faster. One reason for this might be the increase in number of constraints, required by the formalization used. However, we also obtained large solution times for problems that aren't very complex (Case 1) and that require fuzzification of coefficients and, hence, more constraints. Such a situation can be justified by the occurrence of a situation of slow convergence, due to the entrance in a

certain zone of the search space, which is not very favourable, for a certain problem, which leads to a slow solution process of this problem, mainly in the final phase, when the solution quality refinement process occurs.

Making a final general analysis of the results obtained we can verify that for a higher value for Z, i.e. O.F. value, it is generally associated a lower value in terms of global satisfaction of constraints of the corresponding problem (global M), as well as higher time to achieve the result. This is really the essence of the trade-off analysis made in this work.

6 Conclusions

This work shows that the SA algorithm is a good solution technique for solving fuzzy optimization problems. We can summarize its advantages in the following way: simplicity and easy implementation; independence on the problem to be solved (it can solve linear and non-linear problems); it does not require any mathematical transformation of the problem to be solved (as it is the case of the Zimmerman's method).

The main disadvantage of the SA algorithm is the need to define the initial states that satisfy the constraints, for each variable. Another disadvantage is the choice of an appropriate initial temperature value (T), because it can imply a longer search and consequently more computation time. Relatively to the evaluation factor, time, it is convenient to refer that the algorithm's solution time did not exhibit a linear and clearly predictable behaviour, but a behaviour that depended, strongly, on the problem. This nature of convergence is a typical aspect of non-deterministic search methods, such as the SA.

A trade-off analysis was performed both in terms of satisfaction of constraints and time to achieve results versus the objective function value. It should be noticed that, in general, if the decision maker intends to reach better constraints satisfaction level (i.e. coefficients and/or parameters) that will result in worse results in terms of the objective function and vice versa.

In terms of future research it would be interesting to extend this study, through a comparison of the results here obtained with the results for other values for the algorithm's control parameters, namely the initial temperature value and eventually also try to adjust the temperature according to the nature of specific problems. Additionally we can also consider introducing heuristics to facilitate the resolution process of the problems considered. Another interesting aspect for future work would be to evaluate the algorithm's behaviour for the solution of non-linear problems.

References

[1] A. Bundy, "Artificial Intelligence Techniques", Springer-Verlag (1997).

[2] C. Reeves, "Modern Heuristic Techniques for Combinatorial Problems", McGraw-Hill (1995).

[3] E. Aarts, J. Lenstra, "Local Search in Combinatorial Optimization", John Wiley & Sons Inc. (1997).

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

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

[6] H. Zimmermann, "Fuzzy Set Theory and its Applications", Kluwer Academic Publishers, (1991).

[7] J. Blazewics, K. Ecker, E. Pesch, G. Schmidt and J. Weglarz, "Scheduling Computer and Manufacturing Processes", Springer-Verlag (1996).

[8] J. Filho, P. Treleaven, C. Alippi, "Genetic-Algorithm Programming Environments", *IEEE Computer Society*, pp. 28-43 (1994).

[9] L. Zadeh, "Fuzzy Sets: Information and Control", **8**, pp. 338-353 (1965).

[10] M. L. R. Varela, R. A. Ribeiro "Utilização de Simulated Annealing em Optimização Difusa" *Revista de Investigação Operacional*, **21** (2), pp. 205-231 (2001).

[11] R. A. Ribeiro, F. M. Pires, "Fuzzy Linear Programming via Simulated Annealing", *Kybernetica*, **35** (1), pp. 57-67 (1999).

[12] R. Bellman, L. Zadeh, "Decision-Making in a Fuzzy Environment", *Management Science*, **17**, (4), December (1970).

[13] Y. Lai and C. Hwang, "Fuzzy Mathematical Programming - Methods and Applications", Springer-Verlag (1992).

[14] Y. Lai and C. Hwang, "Fuzzy Multiple Objective Decision Making – Methods and Applications", Springer-Verlag (1994).