B A D A N I A O P E R A C Y J N E I D E C Y Z J E

Nr 4 2003

José Luis VERDEGAY*
Edmundo VERGARA-MORENO**

SOLVING *NP*-HARD PROBLEMS BY USING FUZZY SETS-BASED HEURISTICS***

The main aim of this paper is to show how fuzzy sets and systems can be used to produce optimization algorithms able of being applied in a variety of practical situations. Fuzzy sets-based heuristics for Linear Programming problems are considered. To show the practical realisations of the approach proposed a metaheuristic proving the efficiency of using fuzzy rules as termination criteria is shown. Then the Travelling Salesman Problem and the Knapsack Problem are assumed and solved by means of this metaheuristic giving rise in this way to new heuristic algorithms solving these problems. Finally, as an illustration, some practical results showing the outstanding potential of these algorithms are shown.

1. Introduction

It is assumed that generally on the first level, the principal constituents of Soft Computing are the Approximate Reasoning and the Functional Approximation/Randomised Search, and then on the second level Probabilistic Models, Fuzzy Sets and Systems, Evolutionary Algorithms (EA) and Neural Networks appear. On the one hand, it is evident that since the famous "Fuzzy Boom" of the 1990's, Fuzzy Sets and Systems have settled permanently in all the areas of R+D+I. Their applications can be found in all the fields of our daily life, and they are a subject of study on different educational levels. On the other hand, there is no doubt that thanks to

* Department of Computer Sciences and Artificial Intelligence, University of Granada, Daniel Saucedo Aranda s/n, 180071 Granada, Spain, e-mail: verdegay@decsai.ugr.es (corresponding author).

** Department of Mathematics, National University of Trujillo, Av. Juan Pablo II s/n, Trujillo, Perú, e-mail: vergara@unidu.edu.pe.

*** Research supported by the Project Heuristics (TIC2002-4242-CO2-02).

the technological potential that we have nowadays, we tackle witness discoveries that were unpredictable only a decade ago.

In particular, computers perform efficiently tasks that seemed to be very laborious, it not impossible, only a short time ago, allowing us to tackle problems of great complexity, both in comprehension as well as in dimension, in a great variety of fields. Special attention should be paid to the optimization field [1]–[3], [6]. In those fields the EA's appear to be much valuable methods for finding solutions to specific problems. However, EA's are just one more class of heuristics [4], including Taboo Search, Simulated Annealing, Variable Neighbourhood Search (VNS), Hill Climbing, Memetic Algorithms and many others. All of them usually give solutions that are not the very best, but their quality is good enough to satisfy the decision maker. Consequently, among the constituents of Soft Computing, instead of EA's, that might represent only part of the methods of search being used, there should appear Heuristic Algorithms.

But, in spite of the huge success achieved by the Fuzzy Sets and Systems, the important progress produced by the heuristics in a variety of practical ways, and close relationship between both methodologies, if we do not consider the super-area of the Genetic Algorithms, or more generally of the EA, not much work has been done in the development of Fuzzy Sets-based Heuristics [14]–[16].

As an attempt to bridge this gap, the aim of this contribution is to show how fuzzy sets-based methodologies can be used as and aid in solving optimisation problems, thus giving rise to the so-called Fuzzy Sets-based Heuristics. In this way, first we will focus on the possibility of using fuzzy rules as termination criteria in the algorithms. Then the application of this methodology to two classical and very well known problems, the Travelling Salesman Problem and the Knapsack Problem, will be shown and described. Finally, some experimental results will also be analyzed.

2. Using fuzzy rules for terminating algorithms

The key point in this section is that Fuzzy Linear Programming (FLP) methodologies may help to find solutions to problems in which finding an optimum solution is not easy. As is well known, there are a lot of *NP*-hard problems (Knapsack, Travelling Salesman, etc.) which cannot effectively be solved in all cases, but which are of the utmost importance in a variety of real applications. In these problems, the decision-maker must usually accept approximate solutions instead of optimum ones. At this point the aim here is to show how the FLP can help classical MP models and techniques by providing approximate (fuzzy) solutions that may be used by the decision-maker to obtain quickly a solution that would be good enough for these prob-

lems.

Let us justify this fact. An algorithm for solving a general classical optimisation problem can be viewed as an iterative process that produces a sequence of points according to a prescribed set of instructions, together with a termination criterion. Usually, we are interested in algorithms that generate a sequence $x_1, x_2, ..., x_N$ that converges to an overall, optimum solution. But in many cases, however, and because of the difficulties appearing in the problem, we may have to be satisfied with less favourable solutions. Then the iterative procedure may stop at either 1) if a point belonging to a prefixed set (the solution set) is reached, or 2) if some prefixed condition for satisfaction is verified.

But, the conditions for satisfaction are not to be meant as universal ones. In fact they depend on several factors such as the decision-maker, the features of the problem, the nature of the information available, etc. In any case, assuming that a solution set is prefixed, the algorithm will stop if a point in that solution set is reached. Frequently, however, the convergence to a point in the solution set is not easy because, for example, of the existence of local optimum points, and hence we must redefine some rules for terminating the iterative procedure.

Roughly speaking, the possible criteria to be taken into account for terminating the algorithms are nothing but control rules. Thus these rules could be associated with the above two points: the solution set and the criteria for terminating the algorithm. As is clear, fuzziness can be introduced in both points, without assuming it as inherent in the problem, but considering it to be an aid for obtaining, in a more effective way, some solution for satisfaction the decision-maker's wishes. This is meant so that the decision-maker might be more comfortable when obtaining a solution expressed in terms of satisfaction instead of optimisation, as is the case when fuzzy control rules are applied to the control processes. Therefore, and in the particular case of optimisation problems [5], [17]–[19], it makes sense to consider fuzziness:

a) In the Solution Set, i.e., there is a membership function giving the degree with which a point belongs to that set, and

b) On the conditions for satisfaction, and hence Fuzzy Control rules on the criteria for terminating the algorithm.

In the particular case of LP problems, if a conventional problem is assumed

$$\min \{cx/Ax = d; \quad x \geq 0\}$$

the Simplex Algorithm, with the usual denotation, can be summarised as follows:

1) Find an initial extreme point x with basis B,
2) Let x be an extreme point with basis B, and let R be the matrix corresponding to the nonbasic variables. Compute $c_B B^{-1} R - c_R$

   If this vector is non positive then stop, x is an optimal extreme point.

   Else select the most positive component $c_B B^{-1} a_j - c_j$ and compute $y_j = B^{-1} a_j$:

If $y_j = B^{-1}a_j$ is less than or equal to 0 Then stop. Objective unbounded.
If $y_j = B^{-1}a_j$ is neither less than nor equal to 0 Then go to step 3

3) Find the new extreme point by changing the current basis. Repeat step 1.

Therefore, as may be seen, in the Simplex Algorithm control rules appear mainly in the second step as

– The non positivity of the vector $c_B B^{-1}R - c_R$ could be meant in a soft sense,
– The positivity of the component $c_B B^{-1}a_j - c_j$ could be measured according to some membership function, and
– The accomplishment of $y_j = B^{-1}a_j \leq 0$, if this is viewed as a constraint, could be fuzzified.

If the first possibility is considered, a new second step can be formulated,

2′) Let x be an extreme point with basis B. Compute $c_B B^{-1}R - c_R$. If
$\forall j = 1, ..., n$, $c_B y_j - c_j <_f 0$, $c_j \in c_R$
Then stop.

Thus this condition is stated as a fuzzy constraint, in which the decision-maker can accept violations in the accomplishment of the control rules, $c_B y_j - c_j < 0$, to obtain a near, and therefore approximate, optimal solution instead of a full optimal one.

3. Fuzzy control rules in the Travelling Salesman Problem

In this section, the above fuzzy rules, meant as termination criteria in optimization algorithms, will be illustrated by means of a classical and well known problem: the Travelling Salesman Problem (TSP). TSP finds application in a variety of situations: postal routes, tightening the nuts on some piece of machinery on assembly lines, etc. In short, TSP is addressed as follows: Let G be a directed graph in which the nodes represent cities and each edge is assigned a positive cost (the distance between every two cities). If a route of G is defined as a directed cycle that includes every vertex in G, and the cost of a route is the sum of the cost of the edges on the route, the TSP is to find a route of minimum cost.

We denote by $i = 1$ the first city of the route and by 2, 3, ..., n the other cities, d_{ij} is the distance between the city i and the city j, the value of the variable x_{ij} is 1 if j is the next city in the route to city i and 0 otherwise. If $N = \{1, 2, ..., n\}$, the mathematical formulation of the TSP is:

$$\min \quad z = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}$$

$$\text{s. t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,...,n,$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,...,n,$$

$$\sum_{i\in Q}\sum_{j\in \overline{Q}} x_{ij} \geq 1 \qquad \forall Q \subset N$$

$$x_{ij} \in \{0,1\} \qquad i, j = 1,2,...,n.$$

In order to introduce fuzzy termination criteria in the exact algorithms of TSP, we consider that the value of a TSP optimal solution is not a crisp unknown value, but a vague value, because in a large dimension TSP, for which the exact algorithms known need a lot of time to obtain an optimal solution, the decision maker can be comfortable with having an almost optimal solution instead of the very optimal one. In such a situation the optimal value can be seen as a fuzzy set on $[L_0, U_0]$ defined by a membership function as:

$$\mu(z) = \begin{cases} 1 & \text{if} \quad z < L_0 \\ f(z) & \text{if} \quad L_0 \leq z \leq U_0 \\ 0 & \text{if} \quad z > U_0 \end{cases}$$

where $f(z) \in [0, 1]$ $\forall z \in [L_0, U_0]$, is a non-increasing continuous function, L_0 is the lower bound and U_0 the upper bound of the optimal value of the TSP which shall be determined *a priori* as will be shown. As usual, this membership function shows that if the value z of a TSP route is greater than U_0, then it is not allowed by the decision maker. A lower value to L_0 can be a good solution and values between L_0 and U_0 are admissible, but the level of admission will be increasing when z decreases. Obviously, the highest level of admission is obtained when z is equal to L_0.

If the decision maker accepts a non-optimal solution with a membership degree not lower than α ($0 < \alpha < 1$), the termination criterion is:

$$\mu(z) \geq \alpha \quad \text{or} \quad z \leq f^{-1}(\alpha) \tag{1}$$

The values of L_0, U_0 and the function f must be the correct ones in order to provide the expected results by the decision maker. Unsuitable values of L_0, U_0 can produce solutions with great errors. Equally, an incorrect function f can cancel out the flexibility. From [13], [17]–[19] it follows that in such situations a good option is to use a concave function in the definition of the membership function, that is, the function

$$f(z) = \sqrt[n]{\frac{U_0 - z}{U_0 - L_0}} \tag{2}$$

for which the bounds L_0 and U_0 can be computed by the existing suitable and efficient algorithms.

In order to illustrate the use of fuzzy termination criteria in the TSP, the well known algorithm by Little et al. [8], denoted here as LMSK algorithm for short, which was specially designed for solving TSP, has been considered.

3.1. LMSK algorithm

This is a branch and bound algorithm that uses relaxation of TSP as a matching problem denoted by PA (TSP). The algorithm starts by solving the PA (TSP) by the Hungarian Method; if the solution obtained does not possess sub-routes, then it is an optimal solution of the TSP. Else the algorithm proceeds to branch. In each iteration, one chooses the most recent sub-problem TSP_k from among the unsolved ones. If the optimal value is lower than the best current value, then it is saved as the best current value, or alternatively one the branches according to this problem has sub-routes or has not. If the optimal value is equal to or greater than the best current value, then one rejects the sub-problem and starts another iteration. The rule for branching consists in choosing a variable x_{ij} and obtaining two sub-problems by assigning 0 and 1 values to the selected variable. The process terminates when there is not a single sub-problem left unsolved.

In a TSP_k (node k) sub-problem, as a consequence of the above branching, there are variables x_{ij} with fixed values (0 or 1). In graph terms, there are (i, j) edges included or not in the route. We denote by I the included edge set and E the excluded edge set. Then, TSP_k can be described as:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in I} d_{ij} + \sum_{i\in S}\sum_{j\in T} d'_{ij}\, x_{ij} \\
\text{s.a.} \quad & \sum_{i\in S} x_{ij} = 1, \quad \forall j \in T, \\
& \sum_{j\in T} x_{ij} = 1, \quad \forall i \in S, \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j), \quad i \in S, \quad j \in T
\end{aligned}$$

where:

$$S = \{i/(i,j) \notin I \ \ \forall j\}, \quad T = \{j/(i,j) \notin I \ \ \forall i\} \quad \text{and} \quad d'_{ij} = \begin{cases} d_{ij} & (i,j) \notin E \\ \infty & (i,j) \in E \end{cases}$$

furthermore d_{ij} are the coefficients of the matrix of reduced distance of previous node.

This sub-problem is solved by using the Hungarian Method. If the solution obtained has sub-routes one proceeds to branch. A rule for branching is to choose a variable x_{rs} where $r \in S$ and $s \in T$, and to make two nodes with value 1 or 0 assigned to each. Little et al. [8] suggest to select a variable x_{rs} with value 0 if this variable has the maximal potential of increasing in the objective function of the sub-problem. Thus, let

$$\left\{\bar{d}_{ij}\right\} \quad i \in S, \quad j \in T$$

be the reduced cost of the optimum solution of the sub-problem. Then, for each edge (i, j), $i \in S, j \in T$ with reduced cost 0, we compute:

$$p_{ij} = \min\left\{\bar{d}_{ik} / h \in T - \{j\}\right\} + \min\left\{\bar{d}_{hj} / h \in S - \{i\}\right\}$$

which is the minimum amount to increase the optimum value of the assignation to the sub-problem, if the chosen variable is fixed to 0. Therefore, we can choose x_{rs} such that:

$$p_{rs} = \max\left\{p_{ij} / i \in S,\ j \in T,\ \bar{d}_{ij} = 0\right\} \tag{3}$$

when the variable of branching x_{rs} is chosen, all the new nodes can be obtained making x_{rs} =1 and x_{rs} =0. In the first new node, I has the edge (r, s) as the new element, and in the second new node, E has the edge (r, s) as the new element.

3.2. Steps of LMSK algorithm

Step 1: [Starting] Let $U = \infty$ (best bound and real value) and L = {TSP} (subproblem list).

Step 2: [Selecting a sub-problem] If $L = \Phi$ then one terminates the process, because the route associated to U is an optimal one (if $U = \infty$, the TSP has not solution).
If $L \neq \Phi$, one chooses the more recent sub-problem TSP_i, and one removes it from the list L. Go to step 3.

Step 3: [Upper bound determination] Solve PA(TSP_i) by means of the Hungarian Method. Let Z_i be the obtained value.
If $Z_i \geq U$, go to step 2.
If $Z_i < U$ and the solution is a route for TSP (there are not subroutes) then make $U = Z_i$.
If $Z_i < U$ and the solution is not a route for TSP (there are subroutes) go to step 4.

Step 4: [Branching] Choose x_{rs} according to (3) and generate two news sub-problems TSP_{i1} and TSP_{i2} by fixing $xrs = 0$ and $xrs = 1$. Let $L = L \cup \{TSP_{i1}, TSP_{i2}\}$. Go to step 2.

Remark: Note that the termination criterion of this algorithm is $L \neq \Phi$.

3.3. Fuzzy termination criteria in the LMSK algorithm

To introduce a fuzzy termination criterion in the LMSK algorithm, we make a change at the starting step in order to determine the bounds L_0 and U_0. L_0 is computed by using the method proposed in [12], and the upper bound U_0 is computed by means of the process described in [10]. In the same starting step, the decision maker will choose and fix α (the lowest level of admission). Finally, at step 2 one must include the fuzzy termination condition (1). Therefore the following new algorithm is obtained:

Step 1: [Starting] Let $U = \infty$ (best bound and real value) and $L = \{TSP\}$ (subproblem list). Solve by means of the Hungarian Method PA(TSP). If optimum matching is a route of TSP go to step 2. Else, go to 1'.

Step 1': Find L_0 and U_0, then make $U = U_0$ (best real bound) and go to step 1'';

Step 1'': Fix α ($0 < \alpha \leq 1$). If $0 < \alpha < 1$ let $z_0 = f^{-1}(\alpha)$ (bound for the admissible solution, where f is as in (2)). If $L \neq \Phi$ go to step 2. Else, go to step 4.

If $\alpha = 1$ (the decision maker does not want to improve an admissible solution), let $L = \Phi$ and go to step 2.

Step 2: [Selecting a sub-problem] If $L = \Phi$ or $U \leq z_0$ stop the process, as the associated route with U is admissible; if $L \neq \Phi$ go to step 1''. Otherwise stop.

If $L \neq \Phi$ and $U > z_0$, select the more recent problem TSP_i, remove it from the list L and go to step 3.

Step 3: [Upper bound determination] Solve PA(TSP_i) by means of the Hungarian Method. Let Z_i be the obtained value.

If $Z_i \geq U$, go to step 2.

If $Z_i < U$ and the solution is a route for TSP (there are not subroutes) then let $U = Z_i$.

If $Z_i < U$ and the solution is not a route for TSP (there are subroutes) go to step 4.

Step 4: [Branching] Choose x_{rs} according to (3) and generate two news sub-problems TSP_{i1} and TSP_{i2} by fixing $x_{rs} = 0$ and $x_{rs} = 1$. Take $L = L \cup \{TSP_{i1}, TSP_{i2}\}$. Go to step 2.

The introduction of the fuzzy termination criterion on the algorithm has made it more flexible. Now, the decision maker can control the iterations because at step 1'' he can introduce small values for α and to increase them if he wants to improve the admissible solution. Consequently, the decision maker will take into account the time used for obtaining admissible solutions.

For the sake of illustration, let us finally consider the following TSP of 10 cities, with a distance matrix given by:

$$(d_{ij}) = \begin{bmatrix} - & 1 & 62 & 56 & 54 & 27 & 30 & 27 & 55 & 60 \\ 90 & - & 66 & 77 & 52 & 98 & 12 & 55 & 7 & 64 \\ 30 & 41 & - & 60 & 59 & 17 & 72 & 82 & 76 & 21 \\ 57 & 33 & 33 & - & 64 & 78 & 62 & 24 & 70 & 72 \\ 95 & 32 & 69 & 74 & - & 97 & 94 & 92 & 96 & 55 \\ 29 & 25 & 40 & 61 & 25 & - & 27 & 81 & 57 & 94 \\ 98 & 52 & 8 & 11 & 89 & 61 & - & 55 & 91 & 37 \\ 52 & 50 & 90 & 33 & 64 & 86 & 37 & - & 91 & 88 \\ 45 & 83 & 31 & 79 & 70 & 22 & 46 & 18 & - & 91 \\ 96 & 62 & 88 & 9 & 2 & 67 & 64 & 43 & 85 & - \end{bmatrix}.$$

We consider the diagonal elements of the matrix and the distances of excluded edges in the iterations with a value $M = 10\ x\ (\max d_{ij})$. Then, $d_{ii} = M$ and $d_{ij} = M$ if the edge (i, j) is excluded from the possible route. Then solving the problem with the exact algorithm LMSK, one obtains the optimal route "1→ 2→ 9→ 6→ 5→ 10→ 4 →8→ 7→ 3→ 1" with a total distance $z = 218$, after solving 15 sub-problems (original problem included).

On the other hand when using the LMSK algorithm with a fuzzy termination criterion, a function f as (2), $n = 2$, and bounds $L_0 = 208$ and $U_0 = 308$ (at the starting step 1'), the admissible solutions for the different values of α are shown in table 1.

Table 1

Admissible solutions for the example (LMSK algorithm and fuzzy termination criteria)

α	$z_0 = f^{-1}(\alpha)$	Admissible route	Admissible value	Sub-problems solved (*It*)
0.5	283	1-7-3-10-5-2-9-8-4-6-1	258	8
0.8	244	1-8-7-4-3-19-5-2-9-6-1	221	10
0.94	219.64	1-2-9-6-5-10-4-8-7-3-1	218	14

One can observe that for $\alpha = 0.8$, an admissible solution is obtained by solving only 66% of all the sub-problems that the classical algorithm solves. However, the admissible value obtained is very close to the optimum. It is then evident that the saving

in time is upper in comparison with the difference between the admissible value and the optimum value. Furthermore, for $\alpha = 0.94$ one obtains an admissible solution which is the exact one, with less iterations being performed than in the original classical algorithm. The greater the number of cities in the TSP, the more evident these advantages are.

4. Fuzzy control rules in the Knapsack problem

Note that from among all a large variety of possible Knapsack Problems (KP) which can be addressed, we will focus here on the 0-1 KP, which can be mathematically formulated by numbering the objects from 1 to n and introducing a vector of binary variables x_j ($j \in J = \{1, ..., n\}$) such that $x_j = 1$ if object j is selected to be introduced in the knapsack, and $x_j = 0$ otherwise. Then, if p_j is the benefit, or cost, of selecting the object j, w_j its size, or weight, and c the capacity of the knapsack, the problem will be to select, from among all the binary vectors x verifying the constraint of capacity the one which maximizes (alternatively minimizes if the p_j are costs) the objective function defined by the benefits, that is, the problem is stated as follows

$$\max\{\Sigma_{j\in J}\, p_j x_j : \Sigma_{j\in J}\, w_j x_j \leq c;\, x_j \in \{0,1\},\, j \in J\} \tag{4}$$

In the following it will be assumed that the objects are ordered in such a way that one verifies

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \ldots \geq \frac{p_n}{w_n}. \tag{5}$$

and the objective function of the KP, that is, the objective function in (4) will be denoted by z(KP)

4.1. Dantzig's upper bound

An upper bound U for (4) is such that z(KP) $\leq U$. There exist several algorithms for determining upper bounds [9]. For the sake of easiness here we will use Dantzig's upper bound, which can be obtained from the linear programming relaxation of (4), C(KP), that is, from the continuous KP derived from (4) when $x_j \in [0, 1]$, instead of $x_j \in \{0, 1\}$, $j \in J$. When the objects are ordered according to (5), they are consecutively inserted in the knapsack until the first object, s, which does not fit it is found. This object is called "critical object" and it may be defined as

$$s = \min\left\{ j : \sum_{i=1}^{j} w_i > c \right\}. \tag{6}$$

Dantzig's bound is computed from this critical object as

$$U = \lfloor z(C(PM)) \rfloor = \Sigma_{j=1}^{s-1} p_j + \lfloor \bar{c} p_s / ws \rfloor, \tag{7}$$

$$\bar{c} = c - \Sigma_{j=1}^{s-1} w_j \tag{8}$$

where, as usual, $\lfloor x \rfloor$ denotes the largest integer not greater than x.

4.2. Exact algorithms

As was said, there exist several algorithms solving the KP. In [9], a good revision of them can be found. Apart from the well known greedy algorithms, the most relevant methods solving the KP are those based upon either Branch and Bound or Dynamic Programming approach. This is the main reason why in this paper, for illustrating the use of the proposed fuzzy termination criteria, we have focused on a) Horowitz–Sahni algorithm, Branch and Bound based, and b) another Dynamic Programming based algorithm, which in the following will be briefly described.

4.2.1. Horowitz-Sahni algorithm

This is an algorithm which effectively improves the first Branch and Bound approach for the exact solution of KP, formerly presented by P. Kolesar in the 1960's, and basically follows a depth-first methodology by means of which, provided that the objects are sorted as in (5):

1) by the branching scheme at each node, selects the not-yet-fixed object j such that it has the maximum benefit per unit weight. Then creates two descendent nodes by fixing x_j equal to 1 and 0, respectively;

2) the search follows then from the node associated with the insertion of the object j ($x_j = 1$), which basically is a greedy strategy.

The algorithm progresses by doing forward moves and backtracking ones. By the first moves the largest set of new consecutive objects are inserted in the current solution. By means of the second moves it is possible to delete the last inserted object of the current solution. Each time in which a forward move is completed, the upper bound is computed and compared with the best value obtained until that moment. The terminating criterion here is that the algorithm stops when no further backtracking can be performed.

4.2.2. Elimination of states algorithm

From the Dynamic Programming point of view the natural decomposition of a KP will be obtained by considering the sub-problems KP(m, c^{I}), with m ($1 \le m \le n$) objects and a capacity c^{I}, $0 \le c^{\mathrm{I}} \le c$. Let $f_m(c^{\mathrm{I}})$ the optimal solution value of KP(m, c^{I}),

$$f_m(c^{\mathrm{I}}) = \max\left\{ \sum_{j=1}^{m} p_j x_j : \sum_{j=1}^{m} w_j x_j \le c^{\mathrm{I}} \right\} \qquad \text{con } x_j \in \{0,1\}, \quad j = 1, ..., m. \tag{9}$$

To solve the KP by Dynamic Programming this technique considers n stages (the number of possible objects to be selected) and computes, at each stage, the values $f_m(c^{\mathrm{I}})$ of (9) by using the classical way of recursion

$$f_m(c^{\mathrm{I}}) = \begin{cases} f_{m-1}(c^{\mathrm{I}}) \\ \max\left(f_{m-1}(c^{\mathrm{I}}),\ f_{m-1}(c^{\mathrm{I}} - w_m) + p_m \right), \end{cases} \qquad \text{para } c^{\mathrm{I}} = w_m, ..., c. \tag{10}$$

Feasible solutions associated with $f_m(c^{\mathrm{I}})$ values are called states, and the optimal solution of the problem is therefore the state associated with $f_n(c)$. The number of states in each stage, as pointed out in [7], may be reduced by eliminating those states which will not modify at any extent the determination of optimal states. These states are called dominated states. The subsequent algorithm computes and reserves in each step the nondominated states. These states are used as input values in order to find the states of the next stage. Although, as it is patent, the termination criterion of this algorithm is not specified in a concrete way, the process will end when the state $f_n(c)$ is reached.

5. Fuzzy termination criteria for KP solution algorithms

As in many other *NP*-hard problems, KP often accept as optimal solutions good solutions, which, though not being the best ones, are good enough in some sense. To find this kind of good enough solutions, using heuristic algorithms is much appropriated. The alternative way proposed here however, assumes softening the exact algorithms at hand, and solving the particular problem considered by introducing fuzzy termination criteria, which finally provide a new class of heuristic rule-based algorithms.

To illustrate this approach let us consider a KP formulated as in (4), and let us assume an appropriate dimension of it in order to guarantee that it will not be possible to

find its optimal solution in a reasonable time, that is, we suppose that the number of objects is extremely large. In this context two facts are patent:

a) Not any solution can be acceptable.

b) The decision-maker, the problem-solver will always have an imprecise idea (meant as a vague notion) of the range in which he will be able to accept a final value. This imprecise nature is very suitable to be represented by means of a fuzzy set, instead of an interval, as often the decision-maker is very comfortable in expressing his wishes on the values to be accepted if he can also express their corresponding satisfaction degrees, which in essence means that comfortability on the final values to be accepted is a matter of degree.

Coherently let us assume that the fuzzy set representing the proposed decision-maker's satisfaction on the final value to be accepted as a solution is given by means of the following membership function

$$\mu(z) = \begin{cases} 0 & z < L_0 \,, \\ f(t) & L_0 \le z \le U_0 \,. \end{cases} \tag{11}$$

where $f(.)$ is a non decreasing and continuous function, [0, 1] valued and L_0 and U_0 are the lower and upper bounds, respectively, for the optimal value z^* corresponding to the optimal solution, that is, $L_0 \le z^* \le U_0$.

In general, these bounds should be provided by the decision-maker, as he is in charge of defining this acceptability concept. But when the problem has a very high dimensionality, as we suppose here, we need to assume the decision-maker's impossibility to make these data precise. In such a case, very good substitutes of the former vales L_0 and U_0 can be the optimal solution provided by any greedy algorithm applied to (4), for L_0, and Dantzig's bound for U_0. Using these two values in (11), the definition of the membership function may be obtained in a straightforward manner.

Consequently, under these assumptions on bounds and membership function, if the decision-maker is to accept a feasible solution with a value z which has a membership degree greater than or equal to $\alpha \in (0, 1)$, the stop criterion is:

$$f(z) \ge \alpha \quad \text{or} \quad z \ge f^{-1}(\alpha) \tag{12}$$

When this termination criterion is introduced in the algorithms considered here, they are flexibled in such a way that the computing process can be stopped when condition (12) is satisfied.

Provided that the acceptation degree α has been fixed, there are two possibilities as regards the optimal value z^*:

a) It is greater than $f^{-1}(\alpha)$. Then we can obtain an acceptable solution,

b) It is lower than or equal to $f^{-1}(\alpha)$. In this case the termination criterion makes no sense because the process will never end. This is the reason why the exact termination criteria must be maintained, giving thus a subsidiary theoretic character to the

fuzzy termination criterion.

One way to reduce the possibility of situations like b) above is to use, as membership function (11), a concave function, as in such a case

$$h^{-1}(\alpha) < f^{-1}(\alpha)$$

for any function h concave.

A possible concave function which can be used in the definition of the membership function (11) may be the following:

$$f(t) = \sqrt[n]{\frac{t - L_0}{U_0 - L_0}} \tag{13}$$

with $n > 1$. Then (12) becomes:

$$z \geq U_0 + (U_0 - L_0)\alpha^n . \tag{14}$$

To clarify the process, and as an illustration, in the following we will modify the termination criteria of the two algorithms presented in the above section

5.1. New termination criterion for the Horowitz–Sahni algorithm

In the Horowitz–Sahni algorithm, the stopping criterion is stated as to stop the process when there are not any backtracking possibilities. When the algorithm is softened, besides of maintaining this criterion, the condition (12) is also introduced. Thus, with this additional termination criterion the algorithm will analyze each time that a new solution is obtained whether or not the corresponding value satisfies condition (12). In the case of a positive response, the process stops providing the corresponding acceptable solution.

5.2. New termination criterion for the elimination states algorithm

As was said, in this algorithm the termination criterion assumes that the process stops when the state with the value $f_n(c)$ is obtained. According to this criterion the process will be active up to the last state in the last step, although the exact solution is obtained in a previous step or in a state before. By introducing the fuzzy termination criterion in each step and state, it will be analyzed whether an acceptable value has or has not been obtained. In the case of positive response, the process will be stopped. It may appear evident that under this criterion there is not any necessity of solving all the

steps and all the states.

5.3. Experimental results as conclusions

In this section, there will be presented the results of the computational experiments carried out with KP of 1.000, 5.000, 10.000 and 50.000 objects, whose values and weights, in all the cases in the interval [1, 1000], have been randomly generated. Moreover, as usual, in all the problems the capacity of the knapsack is the half-sum of the weights.

Noted that, on the one hand, for each of the algorithms considered in the previous section, with fuzzy termination criteria, a membership degree of $\alpha = 0.5$, $\alpha = 0.8$ and $\alpha = 1$, is successively assumed, being evident that the last case ($\alpha = 1$) corresponds exactly to the classical algorithm with non fuzzy termination criterion. It is also important to remember that in the tables of results below, the error and the execution time are given for each of the above algorithms in section 2, and the error is computed as

Error = 100 x [(Dantzig's Bound) – (Obtained Solution)]/ (Dantzig's Bound)

On the other hand, in order to compare the results obtained, and provided the approximate nature of the algorithms under analysis, we consider Sahni's Algorithm [11], which is a frequently used and therefore known algorithm that runs according to the following greedy scheme: If from among a set of k items M have been already introduced into the knapsack, the algorithm tries to fill up the remaining capacity. To do this, first all the possible sets M consisting of k elements to be introduced in the knapsack are generated. Then the remaining capacity of the knapsack is filled up. For obvious reasons this algorithm is often used for $k = 1, 2, 3$.

Let us finally note that in the fuzzy termination criteria of the two algorithms revised in Section 4, there has been used the condition

$$z \geq L_0 + (U_0 - L_0).\alpha^4$$

that is, the fuzzy criteria are defined by means of the membership function (11) with the function f being given by (13) and $n = 4$; L_0 is the corresponding solution obtained from the basic greedy algorithm, and U_0 is Dantzig's bound.

Table 2

Results from the Horowitz–Sahni Algorithm with fuzzy termination criterion

N	Horowitz–Sahni Algorithm						Sahni Algorithm (approximate)			
	Exact		$\alpha = 0.5$		$\alpha = 0.8$		$K = 2$		$K = 3$	
	error	time	Error	time	error	time	error	time	error	time

1.000	.001928	.044	.00517	.0	.003959	.0	.0032556	.11	.002471	.23
5.000	.000098	1.462	.000464	.0	.000316	.022	.000237	2.538	.000168	4.58
10.000	.00002	3.72	.000108	.01	.000059	.012	.000298	8.802	.000272	13.97
50.000	.00000	21.8380	.000021	.096	.000013	.136	.000003	212.476	.0	257.764

Table 2 shows the results from the experiments after using the Horowitz–Sahni Algorithm with the explained fuzzy termination criteria. In this table, the results that the approximate algorithm of Sahni provides are also shown.

As can be seen, the Horowitz–Sahni algorithm with a fuzzy termination criterion provides results with small errors, and with very short times with respect to both the exact and approximate solutions that Sahni's algorithm reports. This last advantage is much more significant when the number of objects increases.

On the other hand, Table 3 shows other results from the experiments after using the above Elimination of States algorithm with a fuzzy termination criterion. This algorithm is shown to provide a better approximation (smaller error) with shorter times than Sahni's algorithm.

Table 3

Results from the Elimination of States Algorithm with fuzzy termination criterion

N	Elimination of States Algorithm						Sahni Algorithm (approximate)			
	exact		$\alpha = 0.5$		$\alpha = 0.8$		$K = 2$		$K = 3$	
	error	time	error	time	error	time	error	time	error	time
1.000	.001928	.24	.004367	.19	.00382	.236	.0032556	.11	.002471	.23
5.000	.000098	3.778	.000276	3.076	.000247	3.098	.000237	2.538	.000168	4.58
10.000	.00002	12.198	.000069	10.182	.000029	10.252	.000298	8.802	.000272	13.97
50.000	.0	223.944	.000004	218.350	.000003	218.376	.000003	212.476	.0	257.764

Results, as conclusions, given in Table 2 and Table 3 are a proof of the validity of the proposed approach.

Acknowledgement

The first author is indebted to Stefan Chanas for having permitted him to be his friend for the last 20 years of his life. He will never forget him.

References

[1] CADENAS J.M., PELTA D.A., PELTA H.R., VERDEGAY J.L., *Application of Fuzzy Optimization to Diet Problems in Argentinian Farms*, Eur. J. of Oper. Res., 2003, in press.

[2] CADENAS J.M., VERDEGAY J.L., *Optimisation Models with Imprecise Data*, SPUM, 1999, in Spanish.

[3] DELGADO M., KACPRZYK J., VERDEGAY J.L., VILA M.A. (eds.), *Fuzzy Optimisation: Recent Advances*, Physica-Verlag, 1994.
[4] DIAZ A., GLOVER F., GHAZIRI H., GONZALEZ J., LAGUNA M., MOSCATO P., TSENG F., *Heuristic Optimization and Neural Nets*, Ed. Paraninfo, 1996, in Spanish.
[5] HERRERA F., VERDEGAY J.L., *Fuzzy Control Rules in Optimisation Problems*, Scientia Iranica, 1996, 3, 89–96.
[6] HERRERA F., VERDEGAY J.L., *Fuzzy Sets and Operations Research: Perspectives*, Fuzzy Sets and Systems, 1997, 90, 207–218.
[7] HOROWITZ E., SAHNI S., *Computing partitions with applications to the knapsack problem*, Journal of ACM, 1974, 21, 277–292.
[8] LITTLE J.D.C., MURTY K.G., SWEENEY D.W., KAREL C., *An algorithm for the travelling salesman problem*, Operations Research, 1963, 11, 972–989.
[9] MARTELLO S., TOTH P., *Knapsack Problems*, John Wiley and Sons, 1990.
[10] ROSENKRANTZ D.J., STEARNS R.E., LEWIS P.M. II, *An analysis of several heuristics for the travelling salesman problem*, SIAM J. Computing, 1977, 6, 563–581.
[11] SAHNI S., *Approximate algorithms for the 0–1 knapsack problem*, Journal of ACM, 1975, 22, 115–124.
[12] SALKIN H.M., MATHUR K., *Foundations of integer programming*, North–Holland, New York 1989.
[13] SANCHO-ROYO A., VERDEGAY J.L., VERGARA-MORENO E., *Some Practical Problems in Fuzzy Sets-based Decision Support Systems*, Mathware and Soft Computing, 1999, VI, 2–3, 173–187.
[14] VERDEGAY J.L., *Fuzzy Sets Based Heuristics for Optimization*, Springer Verlag, Studies in Fuzziness and Soft Computing Series, 2003.
[15] VERDEGAY J.L., VERGARA-MORENO E., *Fuzzy Termination Criteria in Knapsack Problem Algorithms*, Mathware and Soft Computing, 2000, VII, No. 2–3, 89–97.
[16] VERDEGAY J.L., VERGARA E., *Fuzzy Stop Criteria for Knapsack Problems*, Proceedings of the I Congress of the European Association of Fuzzy Logic and Technologies and del IX Congreso Español sobre Tecnologias y Logica Fuzzy (1999 EUSFLAT-ESTYLF Joint Conference), Palma de Mallorca, 267–270, (in Spanish).
[17] VERDEGAY J.L., VERGARA-MORENO E., *Fuzzy Termination Criteria in Knapsack Problem Algorithms*, Mathware and Soft Computing, 2000, VII, No. 2–3, 89–97.
[18] VERDEGAY J.L., VERGARA-MORENO E., *Fuzzy Sets-based Control Rules for Terminating Algorithms*, Computer Science Journal of Moldova, 2002, 10, 1, 9–27.
[19] VERGARA E.R., *New Termination Criteria for Optimization Algorithms*, Ph.D. Dissertation. Universidad de Granada, 1990 (in Spanish).

Rozwiązywanie *NP*-trudnych problemów przy użyciu heurystyk opartych na zbiorach rozmytych

Głównym celem artykułu jest zaprezentowanie, jak zbiory i systemy rozmyte mogą być użyte w algorytmach optymalizacyjnych, stosowanych w praktycznych problemach. Rozpatrywane są, oparte na zbiorach rozmytych, heurystyki dla zagadnienia programowania liniowego. Dla zaprezentowania praktycznych realizacji zaproponowanego podejścia przedstawiono metaheurystykę, potwierdzającą efektywność stosowania rozmytych reguł jako kryterium zakończenia obliczeń. Za pomocą metaheurystyki rozwiązano takie zagadnienia, jak problem komiwojażera i zagadnienie plecakowe. Na zakończenie przedstawiono wyniki eksperymentów ukazujących nieprzeciętne możliwości proponowanych algorytmów.