Journal of Heuristics, 6: 385–403 (2000)

An Interactive Method for 0-1 Multiobjective Problems Using Simulated Annealing and Tabu Search

M. JOÃO ALVES AND JOÃO CLÍMACO
Faculdade de Economia, Universidade de Coimbra/INESC, Av. Dias da Silva, 165, 3000 Coimbra, Portugal
e-mail: mjoao@inescc.pt

Abstract

This paper presents an interactive method for solving general 0-1 multiobjective linear programs using Simulated Annealing and Tabu Search. The interactive protocol with the decision maker is based on the specification of reservation levels for the objective function values. These reservation levels narrow the scope of the search in each interaction in order to identify regions of major interest to the decision maker. Metaheuristic approaches are used to generate potentially nondominated solutions in the computational phases. Generic versions of Simulated Annealing and Tabu Search for 0-1 single objective linear problems were developed which include a general routine for repairing unfeasible solutions. This routine improves significantly the results of single objective problems and, consequently, the quality of the potentially nondominated solutions generated for the multiobjective problems. Computational results and examples are presented.

Key Words: interactive multiple objective programming, zero-one programming, metaheuristics, simulated annealing, tabu search

The computational complexity of single objective combinatorial problems has provided a basis for an increasing interest in the development of heuristic approaches aimed at dealing with those problems. However, most of these approaches are problem-specific. Therefore one that applies to one problem usually does not to a different one. Over the last decade, several techniques far more generally applicable have been developed such as the following metaheuristic ones: Simulated Annealing (Kirkpatrick, Gellat, and Vecchi, 1983), Tabu Search (Glover, 1986), and Genetic Algorithms (Goldberg, 1989) among others. In spite of being generic heuristic approaches, in most applications reported until now their use has been problem-specific and different implementations have been developed for each new combinatorial problem tackled. The main difficulty of a general purpose algorithm lies in the incorporation of a general mechanism intended to restore the primal feasibility of the system. In fact, it has been shown that the idea of keeping feasibility at every stage of the search is more advantageous than relaxing the constraints and using a penalty function in the cost (Abramson, Dang, and Krishnamoorthy, 1996).

We believe that problem-specific implementations of any metaheuristics should outperform generic ones but, in our opinion, good general-purpose algorithms are be very useful when dealing with problems with no particular structure. This fact motivated our interest in metaheuristics and we started by developing Simulated Annealing and Tabu

Search versions for generic single objective 0-1 linear programs. Until now, few generic versions of metaheuristic methods have been reported. Examples of such approaches are the developments of Aboudi and Jörnsten (1994) and Lokketangen, Jörnsten, and Storoy (1994) for Tabu Search and the Simulated Annealing codes of Connolly (1992) and Abramson, Dang, and Krishnamoorthy (1996).

It is now generally accepted that many real-world problems should consider multiple criteria. Besides, the 0-1 multiobjective problem is a stimulating area of research due to its applications, namely in areas concerning capital budgeting, project selection, delivery and routing, facility location problems and so on. Interactive methods allow the contribution of the decision maker (DM) during the solution search process by inputting information that may lead to solutions more in consonance with his/her preferences. Metaheuristics can play an important role in providing effective interactive processes since the computational effort inherent in exact methods may, depending on the circumstances, jeopardize the interactive process.

The adaptation of metaheuristics to multiobjective problems, even for particular instances, has been little explored so far. However, we should mention the recent work of some researchers in this area: Serafini (1994), Fortemps, Teghem, and Ulungu (1994), Ulungu (1993), Czyzak and Jaszkiewicz (1996) and Hansen (1997).

In this paper we propose an interactive method to solve multiobjective 0-1 linear problems which is based on two metaheuristic approaches: Simulated Annealing and Tabu Search. The basic idea underlying the method is the progressive search of nondominated or “good” approximations of nondominated (potentially nondominated) solutions that belong to regions of interest to the DM. In the first phase of the decision process, the DM may impose bounds on the objective function values (*reservation levels*) that are used to narrow the scope of the search in each interaction. The search of potentially nondominated solutions is made in the area around the solutions that optimize individually each objective function in that restricted region. It provides an approximation of the corresponding nondominated subset that tends to be closer to exact vectors in the extreme areas of the region (where the individual optima for that region lie) rather than in the middle of the region. If the DM wishes to know “better” approximations in the middle, he/she can then specify new reservation levels that narrow the region to be explored with respect to the previous one.

After a global search following metaheuristic routines, the DM may use exact multiobjective techniques that help him/her to come to a final compromise regarding a nondominated solution. These techniques enable the DM to either check if a solution given by a metaheuristics is really a nondominated solution or to compute the nondominated solution(s) closest to a criterion reference point (which represents *aspiration levels* for the objective functions) by optimizing an *achievement* scalarizing function (in the sense of Wierzbicki (1980)). In order to give a better perception of the maxima errors involved in approximations and thus a better evaluation of the quality of those solutions, the DM is provided with bounds given by specific nondominated solutions for the linear relaxation of the multiobjective problem.

The protocol proposed herein to interact with the DM is in accordance with Larichev and Nikiforov (1987) who concluded that, in general, good methods use information relative to the objective function values in the dialogue with the DM in order to create a system of preferences. According to Nakayama (1985) this may lead to two different ways of acting: fixing *aspiration levels* or *reservation levels.*

In Section 1 of this paper we present a version of Simulated Annealing and another of Tabu Search for generic 0-1 linear programs with a single objective. In Section 2 the interactive method for multiobjective problems is discussed and described. Computational examples are presented in Section 3. The paper closes with concluding remarks in Section 4.

1. Simulated annealing and tabu search for single objective 0-1 linear programs

1.1. Simulated annealing

Consider the general scheme of the Simulated Annealing algorithm:

```
Let $S$ denote the solution space, $c$ the objective function to be minimized (cost) and $N$
the neighbourhood structure.
Select an initial solution $x^o \in S$; set $T := T^o > 0$ (initial temperature)
Repeat
    Repeat
        Randomly select $y \in N(x^o)$; $\delta := c(y) - c(x^o)$;
        if $\delta < 0$ then $x^o := y$
        else generate random $d$ uniformly in the range [0,1];
            if $d < e^{-\delta/T}$ then $x^o := y$;
    until $iteration\_count = Nrep$
    set $T := \alpha(T)$;
until $stopping\_condition$ = true;
$x_o$ is the approximation to the optimal solution.
```

T is the control parameter which plays the role of the temperature in the physical system. Initially, with large values of T, large increases in cost will be accepted; as T decreases only smaller increases will be accepted and finally, as the value of T approaches 0, no increases in cost will be accepted at all. The way and the rate at which this parameter is reduced is usually referred to as *cooling schedule*. Detailed descriptions of Simulated Annealing can be found in Kirkpatrick, Gellat, and Vecchi (1983), Aarts and Korst (1989) and Dowsland (1993).

The algorithm given above is a very general one and some decisions must be made in order to implement it: *generic decisions* – T^O, the *stopping_condition* and the cooling schedule (*Nrep* and $\alpha(T)$); *problem specific decisions* – S, c and N.

The Simulated Annealing version we propose here is designed for 0-1 linear programs and will be denoted by *0-1SA* in what follows. Let us begin by the *specific decisions* for *0-1SA.*

A neighbour solution from the current one can be obtained by a *move* which consists in changing the value of one variable x_i from 0 to 1 or from 1 to 0. However, the question of the better way to deal with constraints naturally arises. Concerning specific problems, some authors have proposed relaxing the feasibility conditions including a penalty term in the cost function to discourage violations of the relaxed constraints. We experimented this approach in several types of problems and the results were not attractive. Difficulties arose in the selection of an adequate penalty term. Besides, in highly constrained solution spaces, it is possible that during the whole process no feasible solution is found. The p-median location problem whose formulation is presented in the appendix (entitled P-MEDIAN) is such an example. Although a small problem, no solution was primal feasible during a Simulated Annealing run of 1800 iterations by applying just simple moves like the one stated above. We recall that we are interested in a general algorithm for 0-1 linear programs, so we cannot profit from any special structure of the constraints.

Another possibility is to try to restore the feasibility of each new neighbour solution computed. In this case, the solution space is restricted to the feasible solution space. Connolly (1992) developed a routine to restore solution feasibility called RESTORE which is used in the program GPSIMAN (a general purpose Simulated Annealing). Our computational experiments with this routine led us to realize that a large number of variables must be flipped when the primal feasibility is hardly achieved and the algorithm often backtracks (by undoing the last move), thus requiring high computational effort. We developed a new routine to restore primal feasibility trying to overcome these drawbacks (Alves and Clímaco, 1996). In our opinion, most difficulties with Connolly's routine are due to the computing process of the *most helpful* variable to be flipped. A help-score depends on the way a variable helps violated constraints without regard to how much it can "destroy" satisfied constraints. In our routine, the selection of the *most helpful* variable considers two aspects: how helpful it is for any currently-unfeasible constraint and to what extent it destroys the feasibility of any currently satisfied constraint.

The routine we propose relies on parts of Balas's zero-one additive algorithm (Balas, 1965) and works as follows:

- Initially, all variables are "free" except the one (randomly determined) which was first flipped to obtain the neighbour solution.
  While the solution is not feasible and the feasibility can be restored:
  - find the *most helpful* free variable x_j (if none exists, feasibility cannot be restored)
  - change x_j value from 0 to 1 or the reverse; x_j becomes not free

The *most helpful* free variable x_j is determined in the following manner:

Without loss of generality we assume that all constraints are "$\leq$"

- Calculate the slack s_i for each constraint (satisfied or not): $s_i = RHS_i - LHS_i$
- (constraint i is not satisfied whenever $s_i < 0$)
- Create a subset P of free variables in which one variable is helpful for at least one violated constraint. P is defined as

$$\mathrm{P} = \{j \text{ free: } (a_{ij} < 0 \text{ and } x_j = 0) \text{ or } (a_{ij} > 0 \text{ and } x_j = 1) \text{ for } i \text{ such that } s_i < 0\}$$

  where a_{ij} is the coefficient of variable x_j in the constraint i.
- For every constraint i such that $s_i < 0$, define y_i as

$$y_i = s_i + \sum_{\{j \in P: a_{ij} < 0 \text{ and } x_j = 0\}} (-a_{ij}) + \sum_{\{j \in P: a_{ij} > 0 \text{ and } x_j = 1\}} a_{ij}$$

- If $P = \emptyset$ or there is any $y_i < 0$ then the feasibility of the solution cannot be attained—in this particular case, *0-1SA* will evaluate an unfeasible solution penalizing it in the cost function which means that this process does not backtrack.
  Otherwise calculate the score (≤ 0) of each variable in P. This score measures the weakness of a helpful variable:

$$SCORE_j = \begin{cases} \sum_{\{i:s_i - a_{ij} < 0\}} (s_i - a_{ij}) & \text{if } x_j = 0 \\ \sum_{\{i:s_i + a_{ij} < 0\}} (s_i + a_{ij}) & \text{if } x_j = 1 \end{cases}$$

- The variable with the highest *SCORE* is selected to be flipped.

A zero-score means that the variable is helpful without weakness. Thus, changing the value of this variable yields a feasible solution.

This procedure has performed well in most of the problems tested. However, *0-1SA* gives better results if the selection of the variable to be flipped is random with probabilities proportional to the variable scores instead of selecting the variable with the highest score all the time. This change, which avoids the process to be biased, is only recommended in problems where the feasibility of the solutions is easily restored. *0-1SA* automatically switches from the stochastic to the deterministic choice of the *most helpful* variable when the first iterations show that the feasibility of the problem is not easily restored. This is the case of the P-MEDIAN problem (in appendix).

Generic decisions in 0-1SA

The initial solution: the initial solution is obtained by rounding the optimal solution for the linear relaxation of the problem. If this solution is unfeasible, the procedure described above is used to attain the feasibility.

The cooling schedule: the temperature reduction relation $T := \alpha \,.\, T$ has been considered (usually, $0.8 \leq \alpha \leq 0.99$). T is the control parameter (temperature) and is given by $K.t$. The initial temperature is $T^o = K.t^o$ with t^o being close to 1 and the final temperature is $K.t^f$ with t^f being close to 0. K is automatically determined for each problem. After several experiments using different K values in different problems we have decided to consider $K = 0.5c_{\text{avg}}$ (where c_{avg} denotes the average absolute coefficient in the objective function). The parameters α, t^o, t^f and *Nrep* are specified by the user.

Computational results of 0-1SA

We selected a set of 10 test problems which includes 8 multiple-constraint knapsack problems (MCKP), a p-median problem (in appendix) and a set covering problem (SCP). The MCKP (from 28 to 105 variables) and the SCP (192 variables, 240 constraints) were taken from the literature and are available in the OR-Library by Beasley (1990). The MCKP are the following: PET5, PET6, PET7 which are originally due to Petersen (1967), SENTO1, SENTO2 which are due to Senyu and Toyoda (1967) and WEING6, WEING7, WEING8 which are due to Weingarter and Ness (1967). The SCP is called SCPCYC6 and is from Grossman and Wool (1997). Despite their small size, these MCKP were chosen because they have been tested by other metaheuristic approaches, namely those from Khuri, Bäch, and Heitkötter (1994), Aboudi and Jörnsten (1994), Lokketangen, Jörnsten, and Storoy (1994) and Drexl (1988). The SCP was chosen due to its difficulty. In order to provide a better analysis of the quality of the solution versus time to each problem, we took the computational time of the optimal solution given by the CPLEX commercial solver as the reference time to fix the *0-1SA* parameters. We considered $t^o = 0.98$, $t^f = 0.01$ (for SCPCYC6, $t^f = 0.001$), $\alpha = 0.95$ and *Nrep* was adjusted between 1 and 20 so that a *0-1SA* run would not take longer than 50% of the corresponding CPLEX time. This is of little significance for the MCKP since the maximum CPLEX time is 13 secs for SENTO1. However, concerning the SCPCYC6, owing to the fact that the CPLEX could not find the optimal solution in 10 hours—giving a "better integer value" of 64—we considered then 2 mins 40 secs runs of *0-1SA* that, in fact, provided better solutions. Both CPLEX and *0-1SA* were run in a personal computer Pentium 166 MHz.

Table 1 shows a summary of the computational results produced by 20 runs of *0-1SA* for each problem. It includes the gap between the optimum and the best solution given by *0-1SA* ($100 \times |$optimum-best $|/|$optimum$|$). Concerning the solution quality for the MCKP, some comparisons with other procedures can be made:

(i) *0-1SA* gave a better average solution than the GENEsYs algorithm from Khuri, Bäch and Heitkötter (1994) for all but one (PET6) of the test problems;
(ii) the "best solution" given by *0-1SA* is better than (or equal to) the "best solution" provided by Drexl (1988) or Aboudi and Jörnsten (1994) or Lokketangen, Jörnsten, and Storoy (1994), respectively, for all, for all but one (PET6) and for all but one (PET7) of the test problems.

As stated before, *0-1SA* tries to restore the primal feasibility of each neighbour solution produced by a first random move. Note here that the restoring routine was called at every iteration of the p-median problem (performing nearly 5 additional moves each time) and from 10% (in WEING7) to 70% (in SENTO1 and WEING8) of the iterations of the other problems. The number of additional moves for each solution varied from 1 to 3 within these problems.

Table 1. Computational results of *0-1SA*.

Problem:	P-MEDIAN	PET5	PET6	PET7	SENTO1	SENTO2	WEING6	WEING7	WEING8	SCPCYC6
n*m	20*9	28*10	39*5	50*5	60*30	60*30	28*2	105*2	105*2	192*240
Nrep	1	10	10	6	10	5	5	10	20	20
optimum	3,700	12,400	10,618	16,537	7,772	8,722	130,623	1,095,445	624,319	60
best sol.	3,700	12,400	10,604	16,524	7,772	8,722	130,623	1,095,445	624,319	60
avg sol.	3,700	12,386	10,524.4	16,463.8	7,692.8	8,722	130,235	1,095,352	616,287	65.0
avg time	0.10 sec	0.44 sec	0.39 sec	0.33 sec	3.5 sec	1.27 sec	0.11 sec	0.39 sec	1.9 sec	160 sec
$\text{GAP}^{\text{opt-best}}$	0%	0%	0.13%	0.08%	0%	0%	0%	0%	0%	0%

1.2. Tabu Search

Tabu Search, like Simulated Annealing, is a neighbourhood search heuristics designed to avoid being trapped in local optima. However, in contrast with Simulated Annealing, the randomization is de-emphasized in Tabu Search assuming that intelligent search should be based on more systematic forms of guidance. The search is constrained by classifying certain moves as forbidden (i.e. tabu) in order to prevent the reversal, or sometimes repetitions, of the moves. For more details see Glover (1986, 1989, 1990a, 1990b), Glover and Laguna (1993).

In its simplest form, Tabu Search may be described as follows:

```
Let S be the solution space and $c$ the objective function to be minimized
• Select an initial solution $x^o \in$ S; Let $x^* := x^o$
The tabu list is initially empty: TL := ∅
Repeat
    • Create a candidate list of non-tabu moves—if applied, each move would
    generate a new solution from the current one. So, let $Candidate\_N(x^o)$ be the
    set of candidate neighbour solutions.
    • Choose $y \in Candidate\_N(x^o)$ that minimizes the function $evaluation(y)$ over
    this set.
    If $c(y) < c(x^*)$ then $x^* := y$
    • $x^o := y$ and update TL.
until a specified number of iterations have passed without updating the best solution, $x^*$.
```

The Tabu Search version we propose herein is devoted to 0-1 linear programs (*O-1TS*). A *move* leading to a neighbour solution is defined by changing the value of one variable from 0 to 1 or from 1 to 0. The initial solution is computed as in *0-1SA*.

0-1TS includes three phases: a first phase of search that only uses a list of tabu moves, a second one which is a diversification phase and finally, an intensification phase. Besides short-term tabu memory, which is used throughout the process, frequency-based memories are also included in the diversification and intensification phases. Short-term memory is implemented by a tabu list (last-in-last-out) which records the last #TL (the length of the tabu list) variables changed from 0 to 1 or the reverse. These variables cannot change their values (preventing the reversal of certain moves) unless the *aspiration criterion* is applied. The *aspiration criterion* consists in overriding the tabu status of a move when it yields the best solution obtained up to then.

The diversification phase tries to generate solutions that embody different features (variable values) from those previously found, driving the search into new regions. It uses a frequency memory whose data have been collected from the beginning of the first phase. This memory vector records the number of times each variable took the value 1 in feasible solutions. At the diversification phase, a variable is allowed to change from 0 to 1 (1 to 0) if this move is non-tabu and the variable frequency of 1's is smaller (larger) than a threshold—the threshold is initialized to be the average frequency of 1's of all the variables and is increased (decreased) whenever there is no possible move.

The intensification phase tries to reinforce solution features historically found to be good. It uses two frequency memory vectors whose data are based on solution quality and have been collected from the beginning of the second phase: let x^{bestI} and x^{worstI} be, respectively, the best and the worst feasible solution produced until the end of the first (I) phase; the memory vectors *record1s_good* and *record1s_bad* register the number of times each variable took value 1, respectively, in "good" and "bad" feasible solutions during the second phase; a solution has been considered "good" if $c(x) < c(x^{\text{bestI}})+\delta$ or "bad" if $c(x) > c(x^{\text{worstI}}) - 2\delta$ with $\delta = 0.25[c(x^{\text{worstI}}) - c(x^{\text{bestI}})]$. At the intensification phase, a variable x_i is allowed to change from 0 to 1 (1 to 0) if this move is non-tabu and $\textit{record1s\_good}_i \geq \textit{g1}$ and $\textit{record1s\_bad}_i \leq \textit{b1}$ ($\textit{record1s\_good}_i \leq \textit{g0}$ and $\textit{record1s\_bad}_i \geq \textit{b0}$). Initially, $g1 = \textit{g0}$ and $\textit{b1} = \textit{b0}$ are given by the average value over *record1s_good* and *record1s_bad*, respectively. These thresholds are relaxed whenever there is no possible move. In the current implementation of *0-1TS* the intensification phase is performed 3 times, starting with the three best solutions obtained until the end of the second phase.

0-1TS goes from one phase to the next one after passing ΔN iterations (a number specified by the user) without updating the best solution.

Like in Simulated Annealing, our computational experience showed that repairing unfeasible solutions improve the results remarkably. Therefore, moves that restore feasibility have higher priority. However, we have adopted a different strategy from *0-1SA*: only one move is performed at each iteration yielding a solution, possibly unfeasible, but that surely will move towards feasibility in the next iteration. Taking into account the tabu list, the frequency-based memories, higher priority moves and the aspiration criterion, the set $\textit{Candidate\_N}(x^{\text{o}})$ is defined in different ways whether x^{o} is feasible or not and depending on the search phase. If the current solution x^{o} is feasible, the tabu list, the aspiration criterion and the frequency-based memories (in the last phases of the search) are used to identify the elements included in $\textit{Candidate\_N}(x^{\text{o}})$. The neighbour solution y selected will be the one (not necessarily feasible) that minimizes the function $\textit{evaluation}(y)$ over this set. If x^{o} is not feasible, $\textit{Candidate\_N}(x^{\text{o}})$ only includes solutions obtained by a non-tabu move that reduces the unfeasibility of the solution. According to the restoring routine described above for *0-1SA*, y will be the solution obtained by flipping the highest-score variable. Therefore, the function $\textit{evaluation}(y)$ just evaluates candidate solutions obtained by a move from one feasible solution (the former case). Since these candidate solutions may be unfeasible, $\textit{evaluation}(y)$ must include not only the cost function but also a penalty term that penalizes violated constraints. After experimenting different weights to penalize each violated constraint we realized that lower weights perform better in the first iterations (enabling an oscillation that diversify the search) and higher weights fit quite well in the last iterations (when an intensification is desired). Thus, $\textit{evaluation}(y) = c(y) + W \cdot v$ where v is the number of violated constraints and W is the weight that varies from $c_{\min}$ to $c_{\max}$, the minimum and maximum non-zero absolute coefficients in the objective function, respectively.

Computational results of 0-1TS

0-1TS was applied to the set of problems used for *0-1SA*. The parameter ΔN was specified for each problem so that the computational times were similar to those spent by *0-1SA*. Several lengths of the tabu list, related to the square root of the number of variables, were

Table 2. Computational results of *0-1TS*.

Problem:	P-MEDIAN	PET5	PET6	PET7	SENTO1	SENTO2	WEING6	WEING7	WEING8	SCPCYC6
ΔN	20	70	30	20	50	10	25	20	100	500
#TL	2–6	2–6	2–6	3–7	3–7	3–7	2–6	7–11	7–11	7–13
best sol.	3,700	12,400	10,618	16,499	7,772	8,722	130,623	1,095,445	615,110	62
avg sol.	3,700	12,388	10,588.4	16,499	7,772	8,722	130,389	1,095,409	610,170	62.43
#TL-best	all	6	2; 4	all	all	all	2; 3	7; 8; 9	11	7–11
avg time	0.06 sec	0.44 sec	0.38 sec	0.22 sec	1.37 sec	0.22 sec	0.15 sec	0.5 sec	2.0 sec	158 sec
GAP$^{\text{opt-best}}$	0%	0%	0%	0.23%	0%	0%	0%	0%	1.47%	3.33%

experimented. A summary of the computational results is given in Table 2. This table includes the range of tabu list lengths (#TL) used for each problem and also the best and average solutions among the results produced for the different #TL.

The results of *0-1TS* within our set of test problems showed that variations of #TL do not cause large variations in the solution quality. There is a relative stability in the solutions that usually are good approximations of the optimal solution. In contrast, several runs of *0-1SA* may produce solutions of very different quality due to randomization. Therefore, the average solution of *0-1SA* over several runs is, in general, worse than the average solution of *0-1TS* for different tabu list lengths. Even though, the best *0-1SA* result is often better than or equal to the *0-1TS* result. Thus, we can not conclude that there is a metaheuristics better suited for all the cases than another one.

2. An interactive method for multiobjective 0-1 problems

2.1. Notation and basic concepts

Let $f_i(x)$ be the objective function i to be maximized in the multiobjective 0-1 linear program with k objective functions:

$$\begin{aligned} &\max \ f_i(x) = c^i x \qquad i = 1, \ldots, k \\ &\text{s.t. } x \in S \end{aligned}$$

where $S = \{x \mid Ax = b, x \in \{0, 1\}^n\}$

Let z^o be the image of the feasible solution x^o, so that $z^o = (z_1^o, \ldots, z_k^o) = (f_1(x^o), \ldots, f_k(x^o))$.

The solution z^a *dominates* z^b if and only if $z_i^a \geq z_i^b$ for all objectives $i = 1, \ldots, k$ and $z_i^a > z_i^b$ for at least one objective i.

A solution z is said to be *nondominated* iff there does not exist another one that *dominates* it.

Let us call z a *potentially nondominated* solution (terminology used by Czyzak and Jaskiewicz, 1996) or an *approximation of a nondominated* solution iff the existence of another solution that *dominates* it is still unknown. In what follows, we use the abbreviation p.n.d to designate "potentially nondominated".

2.1. *The interactive method scheme*

The basic idea underlying the method we propose is the progressive and selective search of p.n.d solutions by focusing the search, in each interaction, on a subregion delimited by reservation levels specified by the DM for the objective function values. The use of reservation levels aims at bringing the search process gradually closer to regions of greater interest to the DM. Each computing phase produces a set of p.n.d. solutions more concentrated around the individual optima for the objective functions in the region being explored. Subparts of a region will be explored deeper if the DM specifies new reservation levels further narrowing the previous region. This approach tries to avoid the main drawbacks of generating methods, namely the excessive amount of computational resources required, both in time and storage space, and the large amount of information presented to the DM in each interaction. Thus, the method does not intend to provide a good approximation for the whole set of nondominated solutions, but just for the subregions of greater interest to the DM.

Simulated Annealing and Tabu Search work as two alternative and complementary computing procedures. A computing phase is performed whenever the DM chooses a (sub)region to be explored. It consists in running a metaheuristic routine (Simulated Annealing or Tabu Search) k times (k being the number of objectives). The distinct versions of metaheuristic approaches are adaptations of *0-1SA* and *0-1TS* (described in Section 1) for the multi-objective case. These procedures are similar to single objective ones differing only in the gathering of p.n.d. solutions. A set of p.n.d. solutions, *Set_pnd*, *is* created at the start and updated whenever a feasible neighbour solution (accepted or not) is not dominated by any solution previously found. The neighbour solution is added to the *Set_pnd* removing all the solutions dominated by this new p.n.d solution.

The ith run ($i = 1, \dots, k$) of a metaheuristic routine privileges the objective function i by using standard selection and acceptance criteria. For instance, the probability of accepting the neighbour y of the current solution x^o at the ith run of Simulated Annealing is given by $\min\{1, e^{(f_i(y)-f_i(x^o))/T}\}$ (recall that objective function f_i was defined to be maximized). This implies that, if y dominates x^o or, y is nondominated in relation to x^o and has a better performance than x^o for the objective i, then y is accepted. Otherwise it has a probability of being accepted less than 1. This depends on the magnitude of the decrease in the ith objective function and the temperature T. However, in both cases the set of p.n.d. solutions can be updated with y.

The main reason for so doing—using a pre-defined objective function (i.e. that does not change during the whole run) in each run of Simulated Annealing or Tabu Search—is to avoid damaging the "convergence" character of these metaheuristic routines that could bring out a really good solution to one specific function. An acceptance criterion just based on a dominance/nondominance relation would lead to a more dispersed set of solutions but could risk to be far from all parts of the "true" nondominated frontier. Therefore, since the metaheuristic routine runs k times for each interaction, privileging each objective function individually, the majority of p.n.d solutions are collected nearby the optima for the objective functions in the region under scrutiny. These solutions are usually good approximations for the corresponding nondominated subsets. Although the search is more concentrated on the extreme parts of the region, a relatively dispersed set of p.n.d solutions is obtained due to the unsteady phase of Simulated Annealing for high temperatures and the diversification phase of Tabu Search (which has already been incorporated in *0-1TS*).

This procedure enables the DM to make a strategic search and identify the regions where the solutions corresponding more closely to his/her preferences are placed. Afterwards, a local analysis can provide a better evaluation of the preferred p.n.d solutions or even the computation of "true" nondominated solutions. The interactive method includes different techniques to aid the DM in the final phase of the decision process:

(i) Defining the nondominated frontier of the linear relaxation of the multiobjective problem confined to a fairly enough subregion. In some situations these upper bounds give the DM a better evaluation of his/her preferred p.n.d solutions. Likewise, whenever the differences between the p.n.d solutions and these upper bounds are below certain thresholds specified by the DM, no further search is needed.

(ii) Evaluating whether a satisfactory p.n.d solution is itself a nondominated solution or not. In the negative case, the outcome is a nondominated solution close to the p.n.d solution. The *achievement* scalarizing program ($\mathrm{P}_{\tilde{z}}$) is solved with the satisfactory p.n.d solution playing the role of the reference point $\tilde{z}$.

$$\begin{aligned} \min \quad & \left\{ \alpha - \rho \sum_{i=1}^{k} c^i x \right\} \\ \text{s.t.} \quad & \tilde{z}_i - c^i x \le \alpha \qquad i = 1, \dots, k \\ & x \in S \end{aligned} \tag{$\mathrm{P}_{\tilde{z}}$}$$

(with ρ positive small enough)

(iii) Using aspiration criterion vectors namely vectors that are nondominated for the linear relaxation of the problem. ($\mathrm{P}_{\tilde{z}}$) is solved producing the nondominated solution closest to the aspiration criterion $\tilde{z}$ according to the Tchebycheff (L_∞) metric (Steuer, 1986).

The aspiration criterion vector philosophy is likely to be most useful in later phases of the decision process when the DM is attempting to hit a final solution.

A Pascal-like outline of the interactive algorithm can be stated as follows:

```
• Let Set_pnd := ∅
Repeat
        • Ask the DM to specify reservation levels for all or some objective values:
          $f_i(x) \ge L_i, i \in K \subseteq \{1, \dots k\}$
          For $j$ := 1 to $k$
```

- run *0-1SA* or *0-1TS* adding $f_i(x) \geq L_i,\ i \in K$ to the original set of constraints, considering $(-f_j(x))$ the cost function and updating *Set_pnd* at each iteration $\rightarrow$ new p.n.d. solutions and the solution in this region that optimizes $f_j(x)$ for the linear relaxation of the problem are obtained

Until the search is well focused on the regions of most interest to the DM
While the DM does not find a satisfactory compromise solution

- The DM chooses a feasible subregion $R \subseteq S$ of interest
- Perform a local analysis in R by choosing one or more options among (i)–(iii):
  - (i) define the nondominated frontier of the linear relaxation of R in order to obtain better upper bounds;
  - (ii) compute the nondominated solution closest (L_∞ metric) to a p.n.d solution belonging to R;
  - (iii) specify aspiration criterion vectors, namely using information produced by (i), to compute nondominated solutions in R.

3. Computational examples

In this section we report a computational experiment with two MCKP. These problems result from including another objective function (randomly generated) in two problems that have been used for single objective experiments—PET7 and SENTO1. Although the procedure is not restricted to two objective functions, both examples are biobjective so that graphical displays are used to present the information. The coefficients of the second objective functions are stated in the appendix.

Concerning the first problem, we have also computed all the "true" nondominated solutions. The purpose of this example is to discuss the quality of the p.n.d solutions obtained by Simulated Annealing and Tabu Search. Although the method is not intended to approximate the whole nondominated frontier, we give an indication of the distance between all the p.n.d solutions produced and the "true" nondominated set when the feasible region has been restricted only twice.

The second example is used to illustrate the interactive algorithm.

For space reasons it is not possible to report all computational tests. We will only mention the most significant ones. The pictures are copies (sometimes superposed) of computer displays.

Example 1. PET7_2, 50 variables (0-1), 5 constraints

We run once Simulated Annealing and Tabu Search independently considering the following sequence of reservation levels: 1st interaction: no limitations; 2nd interaction: $f_1(x) \geq 14500$; $f_2(x) \geq 31700$; 3rd interaction: $f_1(x) \geq 15700$; $f_2(x) \geq 30000$;

While Tabu Search does not show any increasing tendency of the computational times when the region is restricted, the times spent by Simulated Annealing increase significantly. This fact is due to the way the routine for restoring feasibility is used in each case: *0-1TS* performs just one move towards feasibility by iteration and *0-1SA* performs, at each iteration, all the moves needed to reach feasibility, leading to more timing consuming iterations in restricted spaces. Therefore, we increased the number of *0-1TS* iterations from the 1st to the 3rd interactions and decreased the number of iterations in *0-1SA* by considering the following parameters: *0-1SA*–$t^o = 0.98$, $t^f = 0.005$, $\alpha = 0.97$ in all the interactions, *Nrep* = 30 in the 1st interaction and *Nrep* = 5 in the 2nd and 3rd interactions; *0-1TS* − #TL = 7, $\Delta N = 250$ in the 1st interaction, $\Delta N = 500$ in the 2nd interaction and $\Delta N = 1000$ in the 3rd interaction.

The results are shown in the graph of figure 1: 18 p.n.d. solutions from Tabu Search and 14 p.n.d solutions from Simulated Annealing, obtained independently. The computational times spent by Tabu Search were 8, 5 and 6 seconds for the 1st, 2nd and 3rd interactions, respectively. The corresponding times for Simulated Annealing are 6, 27 and 11 seconds.

All nondominated solutions for this problem have also been computed and they are displayed in figure 2. The points represented by a cross (+) are nondominated solutions for the linear relaxation that optimize each objective individually for the region delimited

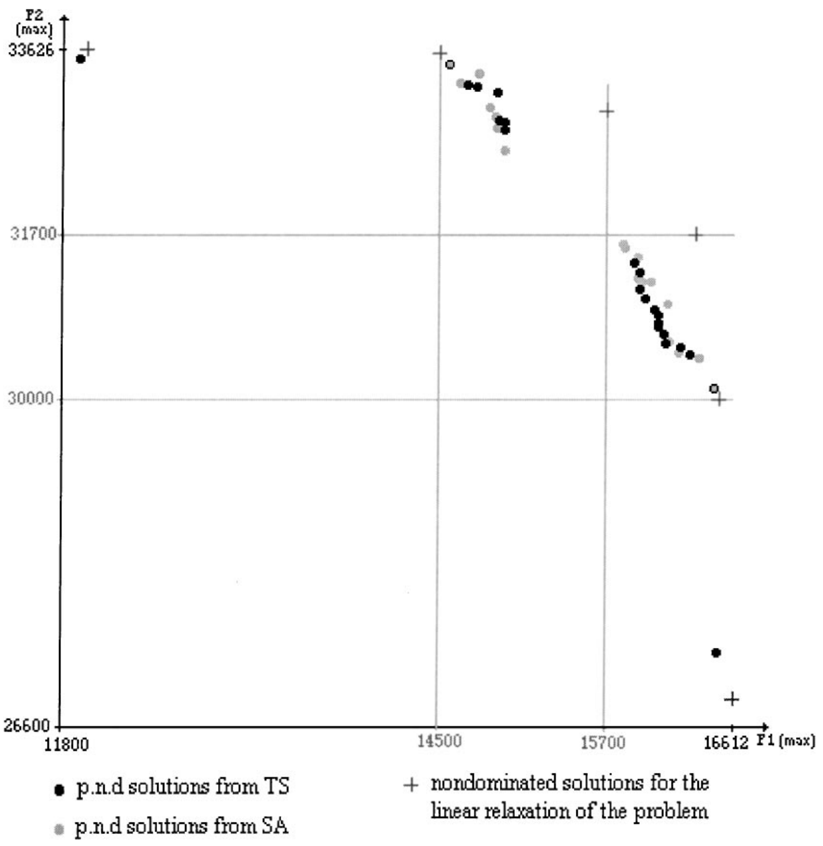

Figure 1. p.n.d solutions for PET7_2 obtained independently from S.A. and T.S.

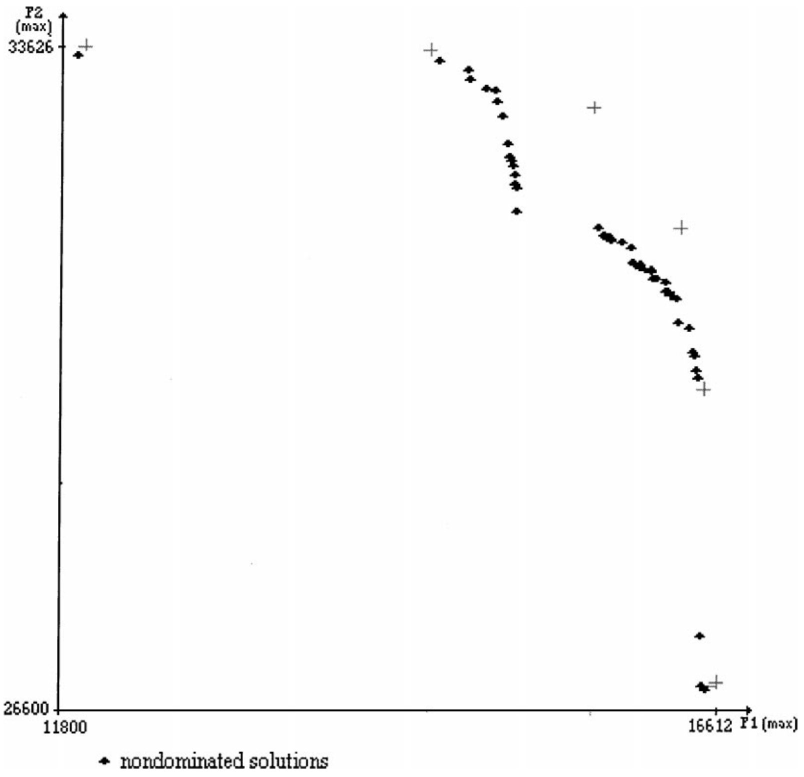

Figure 2. All the nondominated solutions for PET7_2.

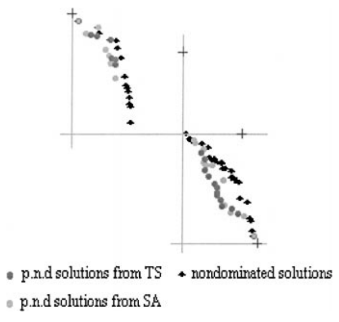

Figure 3. p.n.d solutions for PET7_2 superposed to the nondominated solutions.

in each interaction. Figure 3 superposes graph of figure 1 and figure 2 just in the region where most of the nondominated solutions are placed.

Some quality indices of the p.n.d. solutions may be computed for this problem. If, for instance, the L_∞ metric (maximal component difference) is used to select the nearest nondominated solution, say z^n, from each p.n.d solution z^p, the gap between the two solutions in each dimension may be given by: $100 \times [(z_i^n - z_i^p)/z_i^n]$, $i = 1, 2$. Considering this standard of quality, the following are the average gaps obtained with this experience. Tabu Search: 0.31% in f_1 and 0.19% in f_2. Simulated Annealing: 0.43% in f_1 and 0.20% in f_2.

From our, still limited, experience with this and other multiobjective 0-1 problems, it seems that the tradeoff between solution quality and computational time is better in Tabu Search.

Example 2. SENTO1_2, 60 variables (0-1), 30 constraints

Let us consider that the DM chose the Tabu Search heuristic routine to compute p.n.d solutions with #TL $= 7$ and $\Delta N = 5000$. The outcome of the first interaction, which did not consider additional restrictions, is displayed in figure 4.

Performing two more interactions considering $f_1(x) \geq 5300$, $f_2(x) \geq 5000$ in the 2nd interaction and $f_1(x) \geq 5400$, $f_2(x) \geq 6300$ in the 3rd interaction, a new set of p.n.d. solutions is obtained and is presented here in figure 5.

Let us suppose that solutions 1, 2 and 3, with $z^1 = (6503, 6842)$, $z^2 = (6271, 6878)$, $z^3 = (6250, 7099)$, were the preferred solutions for the DM and he/she wanted to perform a local search in the region delimited by $f_1(x) \geq 6200$ and $f_2(x) \geq 6820$. The DM started by choosing the option (i) that computed the nondominated frontier of the linear relaxation of the problem on the predefined region (figure 6(a)) in order to have a better perception of the largest errors involved with p.n.d solutions. Let us assume that the DM chose then the

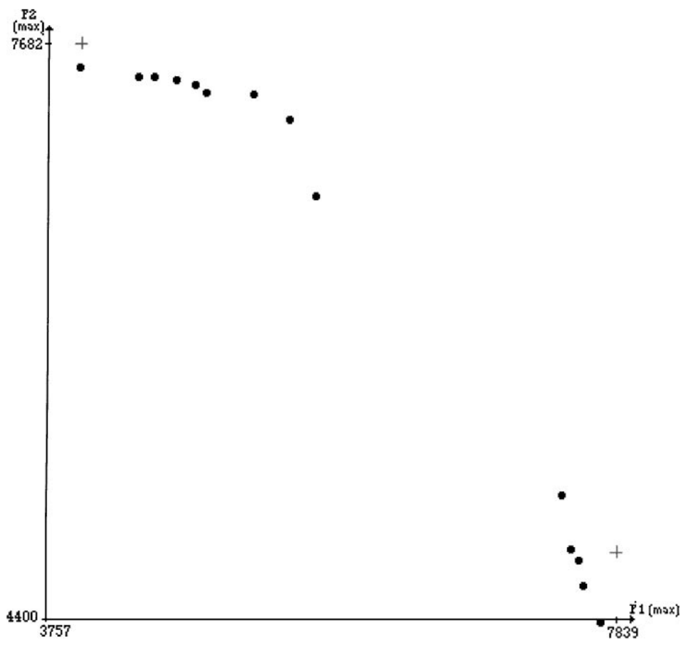

Figure 4. p.n.d solutions for SENTO1_2 obtained in the first interaction.

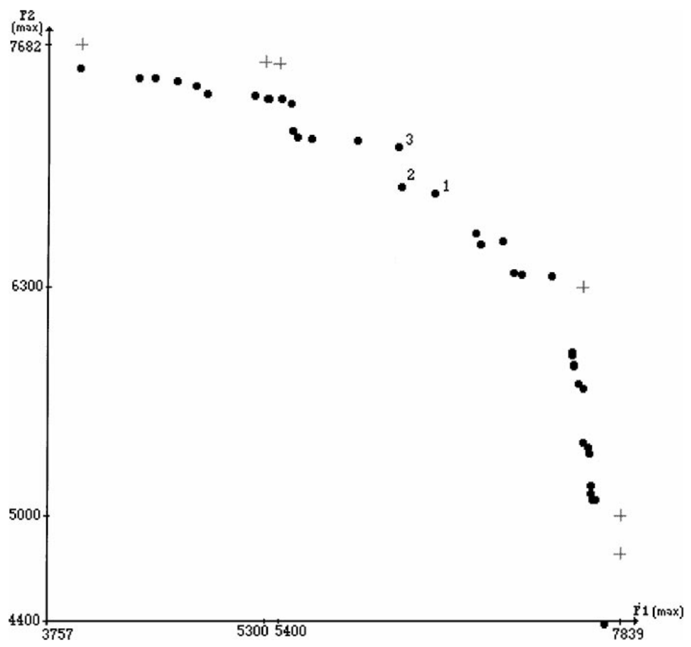

Figure 5. p.n.d solutions for SENTO1_2 after three interactions.

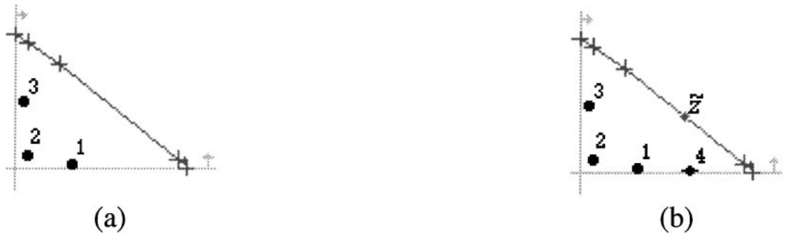

Figure 6. Local analysis.

option (iii) of the algorithm considering the aspiration criterion point $\tilde{z} = (6757, 7056)$. The result (see figure 6(b)) was the nondominated solution $z^4 = (6794, 6828)$.

Note that if the DM had chosen solutions 1, 2 and 3 for option (ii) he/she would have concluded that they are "true" nondominated solutions. It should be stressed that solving $(\mathrm{P}_{\tilde{z}})$ by branch-and-bound to produce exact solutions for this problem (like solution 4) takes a computational time between ten and thirty times larger than the average time spent by one run of Tabu Search.

4. Concluding remarks

This paper presented an interactive metaheuristic algorithm for multiple objective 0-1 linear programs. Two different generic metaheuristic versions — Simulated Annealing and Tabu Search—have been incorporated in an interactive framework. They are alternative and complementary procedures for computing potentially nondominated solutions.

The interactive protocol consists in asking the DM to specify reservation levels for the objective values. This way of interacting with the DM has two main advantages: usually this is the type of information the DM is willing to express in the early stage of the decision process; it enables a progressive reduction of the scope of the search guiding the process towards solution regions that correspond more closely to the DM's preferences.

Some upper bounds for the nondominated solutions are computed and presented to the DM at each interaction. When they are close enough to potentially nondominated solutions, they can be an important element on the decision process since they provide an argument for deciding that there is no need to further explore that subregion.

Exact techniques for computing nondominated solutions have also been included. They are likely to be most useful in the final stage when the DM's preference system is well defined and a satisfactory compromise solution must be selected.

Appendix

P-MEDIAN problem:

$$\begin{aligned}
\min f = {} & 1100x_2 + 1760x_3 + 2200x_4 + 1000x_5 + 1500x_7 + 1000x_8 + 2080x_9 \\
& +1950x_{10} + 650x_{12} + 2400x_{13} + 1200x_{14} + 600x_{15} \\
\text{s.t.}\quad & x_1 + x_2 + x_3 + x_4 = 1 \\
& x_5 + x_6 + x_7 + x_8 = 1 \\
& x_9 + x_{10} + x_{11} + x_{12} = 1 \\
& x_{13} + x_{14} + x_{15} + x_{16} = 1 \\
& y_1 + y_2 + y_3 + y_4 \leq 2 \\
& 110x_1 + 100x_5 + 130x_9 + 120x_{13} - 230y_1 \leq 0 \\
& 110x_2 + 100x_6 + 130x_{10} + 120x_{14} - 230y_2 \leq 0 \\
& 110x_3 + 100x_7 + 130x_{11} + 120x_{15} - 230y_3 \leq 0 \\
& 110x_4 + 100x_8 + 130x_{12} + 120x_{16} - 230y_4 \leq 0 \\
& x_i \in \{0, 1\}\ i = 1, \ldots, 16 \\
& y_i \in \{0, 1\}\ i = 1, \ldots, 4
\end{aligned}$$

Coefficients of the 2nd objective function of the PET7_2 problem:

468	981	484	71	305	392	105	301	141	332	23	495
444	377	458	798	2840	312	151	364	403	138	334	174
3914	78	159	240	41	123	330	500	207	157	348	407
115	3273	102	810	3951	429	137	196	145	3689	3284	477
427	634										

Coefficients of the 2nd objective function of the SENTO1_2 problem:

127	786	27	53	301	40	57	333	488	91	34	56
51	247	65	35	83	5	47	64	831	175	100	17
681	12	27	5	45	92	67	567	43	492	4	280
15	345	44	66	8	99	42	84	74	12	51	698
5	48	2	45	9	52	720	431	53	92	82	20

References

Aarts, E. and J. Korst. (1989). *Simulated Annealing and Boltzman Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley.

Aboudi, R. and K. Jörnsten. (1994). “Tabu Search for General Zero-one Integer Programs using the Pivot and Complement Heuristic.” *ORSA Journal on Computing* 6(1), 82–93.

Abramson, D., H. Dang, and M. Krishnamoorthy. (1996). “A Comparison of Two Methods for Solving 0-1 Integer Programs Using a General Purpose Simulated Annealing Algorithm.” *Annals of Operations Research* 63, 129–155.

Alves, M.J. and J. Clímaco. (1996). “One Experiment Using Simulated Annealing on 0-1 Linear Programming Problems.” *Presented at the Symposium on Combinatorial Optimization*, London, 27-29/3/1996.

Balas, Egon. (1965). “An Additive Algorithm for Solving Linear Problems with Zero-One Variables.” *Operations Research* 15, 517–546.

Beasley, J.E. (1990). “OR-Library: Distributing Test Problems by Electronic Mail.” *Journal of Operational Research Society* 41(11), 1069–1072. See also in WWW site `http://mscmga.ms.ic.ac.uk/info.html`.

Connolly, D. (1992). “General Purpose Simulated Annealing.” *Journal of Operations Research Society* 43(5), 495–505.

Czyzak, P. and A. Jaszkiewicz. (1996). “A Multiobjective Metaheuristic Approach by the Capital Budgeting Model.” *Control and Cybernetics* 25(1), 177–187.

Dowsland, K. (1993). “Simulated Annealing.” In C.R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackweel Scientific Publication, Oxford, ch. 2.

Drexl, A. (1988). “A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack Problem.” *Computing* 40, 1–8.

Fortemps, P., J. Teghem, and E.L. Ulungu. (1994). “Heuristics for Multiobjective Combinatorial Optimization by Simulated Annealing.” *Presented at the XIth International Conference on MCDM*, Coimbra, 1–6/8/1994.

Glover, Fred. (1986). “Future Paths for Integer Programming and Links to Artificial Intelligence.” *Computers and Operations Research* 5, 533–549.

Glover, Fred. (1989). “Tabu Search—Part 1.” *ORSA Journal on Computing* 4(3), 190–206.

Glover, Fred. (1990a). “Tabu Search—Part II.” *ORSA Journal on Computing* 2(1), 4–32.

Glover, Fred. (1990b). “Tabu Search: A Tutorial.” *Interfaces* 20(4), 74–94.

Glover, F. and M. Laguna. (1993). “Tabu Search.” In C.R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackweel Scientific Publication, Oxford, ch. 3.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Mass: Addison-Wesley.

Grossman, T. and A. Wool. (1997). “Computational Experience with Approximation Algorithms for the Set Covering Problem.” *European Journal of Operational Research* 101(1), 81–92.

Hansen, M.P. (1997). “Tabu Search for Multiobjective Optimization: MOTS.” Presented at the *XII International Conference on MCDM*, University of Cape Town, 6–10/1/97.

Kirkpatrick, S., C.D. Gellat, and M.P. Vecchi. (1983). “Optimization by Simulated Annealing.” *Science* 220, 671–680.

Khuri, S., T. Bäck, and J. Heitkötter. (1994). “The Zero-One Multiple Knapsack Problem and Genetic Algorithms.” In E. Deaton, D. Oppenheim, J. Urban, and H. Berghel (eds.), *Proceedings of the 1994 ACM Symposium on Applied Computing*, ACM-Press, New York, 188–193.

Larichev, O. and A. Nikiforov. (1987). “Analytical Survey of Procedures for Solving Multicriteria Mathematical Programming Problems.” In Y. Sawaragi, K. Inoue, and H. Nakayama (eds.), *Towards Interactive and*

Intelligent Decision Support Systems. Springer-Verlag, pp. 95–104. Lecture Notes in Economics and Mathematical Systems, Vol. 285.

Lokketangen, A., K. Jörnsten, and S. Storoy. (1994). “Tabu Search within a Pivot and Complement Framework.” *International Transactions in Operations Research* 1(3), 305–316.

Nakayama, H. (1985). “On the Components in Interactive Multiobjective Programming Methods.” In M. Grauer, M. Thompson, and A. Wierzbicki (eds.), *Plural Rationality and Interactive Decision Processes*, Springer-Verlag, pp. 234–247. Lecture Notes in Economics and Mathematical Systems, Vol. 248.

Petersen, C.C. (1967). “Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R&D Projects.” *Management Science* 13(9), 736–750.

Senyu, S. and Y. Toyada. (1967). “An Approach to Linear Programming with 0-1 Variables.” *Management Science* 15B, 196–207.

Serafini, P. (1994). “Simulated Annealing for Multi Objective Optimization Problems.” In G.H. Tzeng, H.F. Wang, U.P. Wen, and P.L. Yu (eds.), *Multiple Criteria Decision Making, Proceedings of the Xth International Conference*, Taipei 19-24/7/92, Springer-Verlag, pp. 283–292.

Steuer, R. (1986). *Multiple Criteria Optimization: Theory, Computation and Application*. Wiley.

Ulungu, E.L. (1993). *Optimisation Combinatoire Multicritère: Détermination de l’ensemble des solutions efficaces et méthods interactives*. Phd dissertation, University de Mons-Hainaut.

Weingarter, H.M. and D.N. Ness. (1967). “Methods for the Solution of the Multi-Dimensional 0/1 Knapsack Problem.” *Operations Research* 15, 83–103.

Wierzbicki, A. (1980). “The Use of Reference Objectives in Multiobjective Optimization.” In G. Fandel, and T. Gal (eds.), *Multiple Criteria Decision Making. Theory and Application*. Springer-Verlag, Berlin, pp. 468–486. Lecture Notes in Economics and Mathematical Systems, Vol. 177.