Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 169 (2006) 373–391
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

A scatter search method for bi-criteria {0,1}-knapsack problems

Carlos Gomes da Silva [a,b,*], João Clímaco [b,c], José Figueira [b,c,d]

[a] Escola Superior de Tecnologia e Gestão de Leiria, Morro do Lena, Alto Vieiro, Leiria 2401-951, Portugal
[b] Instituto de Engenharia de Sistemas e Computadores de Coimbra, Rua Antero de Quental, 199, Coimbra 3000-033, Portugal
[c] Faculdade de Economia da Universidade de Coimbra, Av. Dias da Silva, 165, Coimbra 3004-512, Portugal
[d] DIMACS Center-Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 108855-8018, USA

Received 27 June 2003; accepted 24 April 2004

Abstract

This paper presents a scatter search (SS) based method for finding a good approximation of the non-dominated frontier for large size bi-criteria {0, 1}-knapsack instances. The method follows the usual structure of SS: (1) diversification, (2) improvement, (3) reference set update, (4) subset generation, and (5) solution combination. For each component specific procedures were built which strongly benefit from the single criterion problem, and also from the characteristics of the neighbourhood of bi-criteria non-dominated solutions. These aspects permit the guidance and restriction of the exploration of the decision space. Experiments were carried out on a large set of instances and the computational results are presented. The approach seems to be very efficient and the quality of the approximation is quite good. These points are also discussed in the paper.

Keywords: Metaheuristics; Scatter search method; Bi-criteria knapsack problems; Combinatorial optimization

1. Introduction

The single constraint {0, 1}-knapsack problem with several criteria is an important issue in the class of multiple criteria combinatorial optimization. Determining the whole set of the non-dominated solutions for large size bi-criteria instances is a very difficult task, despite the recent development of several new techniques, involving both exact (Visée et al., 1998; Captivo et al., 2003) and metaheuristic approaches

* Corresponding author. Tel.: +351 244 820 300; fax: +351 244 820 301/821 301.
E-mail addresses: cgsilva@estg.iplei.pt (C. Gomes da Silva), jclimaco@inescc.pt (J. Clímaco), figueira@fe.uc.pt, figueira@dimacs.rutgers.edu (J. Figueira).

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.08.005

(Czyzak and Jaskiewicz, 1998; Ben Abdelaziz et al., 1999; Gandibleux and Fréville, 2000; Gandibleux et al., 2001).

Exact methods are very limited in dealing with the problem. It is only possible to determine the set of exact non-dominated solutions for random instances of about 500 items (Visée et al., 1998; Captivo et al., 2003). This is mainly due to the computational and memory requirements. In fact, has not yet been developed a method that efficiently generates all the non-dominated solutions.

The approximate methods have to take into account a good compromise between the quality of the approximation and the computational time taken, especially for large size instances. Over the last years were proposed, in different contexts, evolutionary algorithms which enhance that compromise by considering a hybrid approach, avoiding thus pure random search strategies: Ishibuchi and Murata (1998), Zitzler and Thiele (1999b), Murata et al. (2000), Deb and Goel (2001), Morita et al. (2001), Talbi et al. (2001), Balicki and Kitowski (2001), Knowles and Corne (2000) and Jaszkiewicz (2001, 2002a,b, 2003).

The purpose of this paper is to apply the scatter search (SS) methodology to generate a set of solutions that approximates the "non-dominated frontier" for large size bi-criteria knapsack instances. As a population based method, SS is potentially suitable for tackling problems with several criteria. However, the use of this methodology in the multiple criteria field is very rare. There are only a few references to it in the literature: Beausoleil (2001), presented a multiple criteria framework for a scheduling problem, Corberán et al. (2002), considered the problem of routing school buses, and Martí et al. (2000), considered the proctor assignment problem.

The rest of the paper is organized as follows: Section 2 presents the basic definitions and empirical properties of the solutions of the {0, 1}-knapsack problem; Section 3 is devoted to the SS method; Section 4 describes the computational experiments and results; and Section 5 presents the main conclusions and future research.

2. Basic definitions and empirical properties of solutions

The bi-criteria knapsack problem can be formulated as follows:

$$\begin{aligned} \max \quad & z_1(x_1,\ldots,x_n) = \sum_{j=1}^{n} c_j^1 x_j, \\ \max \quad & z_2(x_1,\ldots,x_n) = \sum_{j=1}^{n} c_j^2 x_j, \\ \text{s.t.} \quad & \sum_{j=1}^{n} w_j x_j \leqslant W, \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n, \end{aligned} \tag{1}$$

where c_j^i represents the value of item j on criterion i, $i = 1, 2$, $x_j = 1$ if item j ($j = 1, \ldots, n$) is included in the knapsack and $x_j = 0$ otherwise, w_j means the weight of item j and W is the overall knapsack capacity. We assume that c_j^1, c_j^2, W and w_j are positive integers and that $w_j \leqslant W$ with $\sum_{j=1}^{n} w_j > W$. Constraints $\sum_{j=1}^{n} w_j x_j \leqslant W$ and $x_j \in \{0, 1\}, j = 1, \ldots, n$, define the feasible region in the decision space, and their image when using the criteria functions z_1 and z_2 define the feasible region in the criteria space.

A feasible solution, x, is said to be *efficient* if and only if there is no feasible solution, y, such that $z_i(x) \leqslant z_i(y)$, $i = 1, 2$, and $z_i(x) < z_i(y)$ for at least one i. The image, in the criteria space, of an efficient solution is called a *non-dominated* solution.

The aim is to generate an approximation of the set of the non-dominated solutions. These approximate solutions will be called potentially efficient/non-dominated solutions (PNDS).

It is well known that some non-dominated/efficient solutions (designated supported non-dominated/efficient solutions) can be obtained by using weighted sums of the criteria (Steuer, 1986). Each weighted sum

function is associated to a single criterion knapsack problem. This has been studied extensively by several authors (see, for example, Martello and Toth, 1990; Pisinger and Toth, 1998).

For the single criterion knapsack problem we have the well known empirical property, which is commonly observed in random instances:

Property 1. *The optimal solution of the integer knapsack problem only differs from the continuous solution (that is, when the integrality of variables is relaxed) in a very small number of variables that are close to the fractional one.*

The best known algorithms for the knapsack problem are based on this property. The SS proposed method uses Property 1 in order to search for "good" solutions, desirably non-dominated ones, near the optimum of a weighted sum function.

The following property comes from empirical conclusions that we were able to elicit from the set of efficient solutions corresponding to instances of the bi-criteria knapsack problem. We have considered problems with 100, 200 and 300 items; and for each problem size 15 instances were generated. Coefficients were randomly and uniformly generated within the range [1, 1000]. The capacity of the knapsack is equal to fifty percent of the sum of the weights. The analysis aimed to determine the nearest solution according to the Hamming distance for each efficient solution. That is, for an efficient solution, x^i, a different efficient solution, x^j, was sought, such that $\|x^j - x^j\| = \min\{\sum_{t=1}^{n} | x_t^i - x_t^k |, x^k \in \{\text{efficient solutions set}\}, k \neq i\}$. This distance corresponds to the minimum number of variables to be changed in solution x^i to obtain x^j. Results presented in Table 1, concern the average, the maximum and the minimum Hamming distances.

For each instance, the frequency of the number of variables changed when moving from one to another efficient solution corresponding to the above calculations was also computed. The results are given in Table 2.

Supported by the results obtained, the average of the minimum distance from each efficient solution, of a 100 items instance, is 2.331 variables (2.415 for $n = 200$ and 2.472 for $n = 300$). This means that it is only necessary to change, on average, the value of 2.331 variables (2.415 for $n = 200$ and 2.472 for $n = 300$) in each solution to obtain another efficient solution. The maximum distance is, on average, 4.2 (4.533 for $n = 200$ and 4.867 for $n = 300$). The maximum distances are small, and only represent a small proportion. In fact, for all the considered instances more than 60% of the non-dominated solutions can be obtained from another solution simply by changing a pair of variables. This percentage increases to more than

Table 1
Distances between efficient solutions

n	Average				Max				Min			
	Average	Max	Min	STD	Average	Max	Min	STD	Average	Max	Min	STD
100	2.331	2.500	2.170	0.106	4.200	5.000	3.000	0.542	2.000	2.000	2.000	0.000
200	2.415	2.560	2.300	0.083	4.533	5.000	4.000	0.499	2.000	2.000	2.000	0.000
300	2.472	2.560	2.370	0.062	4.867	6.000	4.000	0.499	2.000	2.000	2.000	0.000

Table 2
Frequencies of the changed items

Items changed	$n = 100$				$n = 200$				$n = 300$				
	2	3	4	5	2	3	4	5	2	3	4	5	6
Average	0.720	0.236	0.042	0.008	0.647	0.291	0.060	0.002	0.611	0.308	0.080	0.002	0.001
Max	0.894	0.439	0.154	0.010	0.754	0.451	0.109	0.003	0.720	0.395	0.122	0.004	0.001
Min	0.533	0.041	0.000	0.006	0.495	0.192	0.022	0.002	0.538	0.194	0.044	0.000	0.001
STD	0.097	0.105	0.036	0.002	0.078	0.080	0.021	0.001	0.058	0.061	0.020	0.001	0.000

90% if we change 2 and 3 variables (Table 2). As the size of instances increases, an increasing in the average number of variables changed is also expected, but the value will be certainly very low in comparison with that size. So, it can be established:

Property 2. *An efficient solution can be obtained from another efficient solution by complementing* (x_j *is changed to* $1 - x_j$) *the value of a small number of variables.*

A consequence of this property is that the use of local search procedures around efficient solutions will be very effective in finding new efficient solutions. Property 2 is also important because it can be used together with Property 1 to shorten the decision space exploration. The SS method presented here, is thus based on the above fundamental properties.

3. A scatter search based method

The proposed SS method is organized according to the usual structure of SS method: (1) diversification method, (2) improvement method, (3) reference set update method, (4) subset generation method, and (5) solution combination method. In this section the specific procedures used in each component are presented.

3.1. The diversification method

The initial set of trial solutions is obtained by computing the entire set of extreme efficient solutions for the relaxed problem:

$$\begin{aligned} \max \quad & z_1(x_1,\ldots,x_n) = \sum_{j=1}^{n} c_j^1 x_j, \\ \max \quad & z_2(x_1,\ldots,x_n) = \sum_{j=1}^{n} c_j^2 x_j, \\ \text{s.t.} \quad & \sum_{j=1}^{n} w_j x_j \leqslant W, \\ & x_j \in [0,1], \quad j = 1,\ldots,n. \end{aligned} \tag{2}$$

The set of extreme efficient solutions of (2) can be rapidly obtained by using the bi-criteria simplex method with bounded variables. We start from the optimal solution of criterion z_2 (if this solution is unique it is certainly efficient; if not, it is necessary to consider the one with the highest value in the criterion z_1 from the alternative solutions). We then proceed with the efficient pivoting described below, until the optimal solution of z_1 is achieved (analogously, if this solution is not unique it is necessary to consider the one with the highest value in the criterion z_2 from the alternative solutions).

The optimum of criterion z_2 is obtained using a greedy procedure that includes the items ordered according to non-increasing values of the profit-to-weight ratio $(\frac{c_j^2}{w_j})$ until the capacity is achieved. With the items ordered in this way, let f be the critical item: $f = \min\{k : \sum_{j=1}^{k} w_j > W\}$. So the solution that maximizes z_2, $x^* = (x_1^*,\ldots,x_j^*,\ldots,x_n^*)$, can be written as follows: $x_j^* = 1$ if $j = 1,\ldots,f-1; x_j^* = \frac{W - \sum_{k=1}^{f-1} w_k}{w_j}$ if $j = f$ and $x_j^* = 0$ if $j = f+1,\ldots,n$.

The variable x_f is the only one in the initial basis, known as the working basis, which can be identified in the simplex method tableau. In this tableau, the reduced prices according to criteria z_1 and z_2 are $(c_j^1 - z_j^1) = c_j^1 - c_f^1 \frac{w_j}{w_f}$ and $(c_j^2 - z_j^2) = c_j^2 - c_f^2 \frac{w_j}{w_f}$, respectively. From x^*, an adjacent efficient solution can be obtained by entering the basis a non-basic efficient variable, x_j, which does not degrade the value of criterion z_1, i.e., a variable for which the following conditions are fulfilled:

(1) there exists a *lambda* such that $0 < \lambda < 1$;
(2) $\lambda(c_k^1 - z_k^1) + (1-\lambda)(c_k^2 - z_k^2) \geqslant 0$ if $x_k = 1 (k \in \{1, \ldots, n\} \setminus \{f\})$;
(3) $\lambda(c_k^1 - z_k^1) + (1-\lambda)(c_k^2 - z_k^2) \leqslant 0$ if $x_k = 0 (k \in \{1, \ldots, n\} \setminus \{f\})$;
(4) $\lambda(c_j^1 - z_j^1) + (1-\lambda)(c_j^2 - z_j^2) = 0$;
(5) $c_j^1 - z_j^1 \geqslant 0$.

The efficient pivoting process, which assures the transition from one extreme efficient solution to an adjacent one, is given below (see, for instance, Steuer (1986) regarding the general case):

1. *Leaving variable candidate*: x_f is the only candidate to leave the working basis.
2. *Entering variable candidate*: Compute the reduced prices $(c_j^1 - z_j^1)$ and $(c_j^2 - z_j^2)$ for all the non-basic variables. The candidate variable entering the basis, x_j, is that which corresponds to the maximum value of $\frac{-c_j^2 + z_j^2}{(c_j^1 - z_j^1) - (c_j^2 - z_j^2)}$ in the following situations:
   (a) $x_j = 1$ and $(c_j^1 - z_j^1) - (c_j^2 - z_j^2) < 0$ and
   (b) $x_j = 0$ and $(c_j^1 - z_j^1) - (c_j^2 - z_j^2) > 0$.

   If the maximum value of $\frac{-c_j^2 + z_j^2}{(c_j^1 - z_j^1) - (c_j^2 - z_j^2)}$ is greater than 1 then it is not possible to increase the value of criterion z_1, hence all the efficient solutions have been found. If not, we proceed by updating the values of the entering and leaving candidates. Due to the use of bounded variables this process is a rather more complicated than the usual one (see, for example, Bazaraa et al. (1992) regarding the general case). In the knapsack problem we have:
3. *Updating the value of variable x_j (candidate to entering the basis)*:
   (a) If $x_j = 0$, then the new value of x_j is $\min\{x_f \frac{w_f}{w_j}, 1\}$. In this case, we have:
      (a.1) If $\min\{x_f \frac{w_f}{w_j}, 1\} = 1$, then x_j remains non-basic but its value is now its upper bound. Otherwise,
      (a.2) x_j enters the basis in substitution of x_f.
   (b) If $x_j = 1$, then the new value of x_j is $1 - \min\{(1 - x_f)\frac{w_f}{w_j}, 1\}$. In this case, we have:
      (b.1) If $\min\{(1 - x_f)\frac{w_f}{w_j}, 1\} = 1$, then x_j remains non-basic but its value is now its lower bound. Otherwise,
      (b.2) x_j enters the basis in substitution of x_f.
4. *Updating the value of variable x_f (candidate to leave the basis)*: If x_f becomes a non-basic variable, its value is determined taking into account the value of the new basic variable:
   (a) If $x_j = 0$ and $\min\{x_f \frac{w_f}{w_j}, 1\} = x_f \frac{w_f}{w_j}$, then $x_f = 0$;
   (b) If $x_j = 1$ and $1 - \min\{(1 - x_f)\frac{w_f}{w_j}, 1\} = 1 - (1 - x_f)\frac{w_f}{w_j}$, then $x_f = 1$.

   Nevertheless the variable x_f can remain basic if x_j changes from its upper bound to its lower bound or *vice versa*. In this case, x_j remains non-basic and the new value of x_f becomes $x_f - \frac{w_j}{w_f}$ if x_j changes from its lower to the upper bound, or $x_f + \frac{w_j}{w_f}$ (if x_j changes from its upper to its lower bound).

The process is repeated, obtaining a new extreme efficient solution, until the optimum of criterion z_1 is achieved. Each solution corresponds to the optimum of a weighted sum of the criteria, which is, in general, close to the integer optimum solution. In fact, let $h(x, \lambda) = \lambda z_1(x) + (1 - \lambda) z_2(x)$, $\lambda \in [0, 1]$, be a weighted sum function. Then, if the items are ordered according to non-increasing values of the ratio $\frac{\lambda c_j^1 + (1-\lambda)c_j^2}{w_j}$ and being $f_\lambda = \min\{k : \sum_{j=1}^{k} w_j > W\}$ and $\bar{w} = W - \sum_{j=1}^{f_\lambda - 1} w_j$, then $h(x, \lambda)$ is bounded by

$$\sum_{j=1}^{f_\lambda - 1} (\lambda c_j^1 + (1-\lambda)c_j^2) \leqslant h(x, \lambda) \leqslant \sum_{j=1}^{f_\lambda - 1} (\lambda c_j^1 + (1-\lambda)c_j^2) + \frac{\bar{w}}{w_{f_\lambda}} (\lambda c_{f_\lambda}^1 + (1-\lambda)c_{f_\lambda}^2). \tag{3}$$

The gap, $g(\lambda)$, for the integer optimal solution of $h(x, \lambda)$ is bounded from above by $\frac{\bar{w}}{w_{f_\lambda}}(\lambda c^1_{f_\lambda} + (1-\lambda)c^2_{f_\lambda}) \leqslant \lambda c^1_{f_\lambda} + (1-\lambda)c^2_{f_\lambda} \leqslant \max\{c^1_{f_\lambda}, c^2_{f_\lambda}\}$, i.e.,

$$g(\lambda) \leqslant \max\{c^1_{f_\lambda}, c^2_{f_\lambda}\}. \tag{4}$$

In the proposed method, the efficient extreme solutions of (2) are considered to be the "seed" solutions for the generation of all others. As shown above, they have the property of being already very close to the integer optimum of the weighted sum functions and, unless the extreme points of (2) are biased, which is very uncommon, the initial set contains a collection of diverse and "good" solutions that span the location of the efficient solutions of (1). These solutions will also be very useful in evaluating the quality of the obtained solutions.

After determining all the efficient solutions of (2), the items are ordered such that $w_1 \leqslant w_2 \leqslant \cdots \leqslant w_n$. This order is used because it is independent from the value of the items on criteria z_1 and z_2 and can avoid an exhaustive analysis of all the items when considering those to be included in the knapsack. With this order, if an item does not fit into the knapsack, the forward items can then be discarded.

3.2. The improvement method

This method aims to restore the feasibility of solutions from the diversification method and to enhance these solutions and those obtained from the combination method (described in Section 3.5). When a solution of (2) is not feasible because it has a fractional variable, the generation of new solutions is investigated which sets to 0 or 1 that variable. In the first case, the solution obtained is clearly feasible, but the remaining capacity may be large enough to include additional items. Two new solutions are created using two heuristic procedures. The first (second) one, fills the knapsack, as much as possible, with the items which have the highest profit-to-weight ratio, according to criterion z_1 (z_2). We call this process a positive bidimensional improvement of the solution. The overview of this positive bidimensional improvement procedure of a solution x^k with fractional item f is given below.

Positive bidimensional improvement (x^k, f)

FOR $i = 1$ TO 2 DO {criteria number}
BEGIN
  $N_0 \leftarrow \{j : x^k_j = 0, j = 1, \ldots, n\} \setminus \{f\}$
  $x^k_f \leftarrow 0$
  $\bar{w} \leftarrow W - \sum_{j=1}^{n} w_j x^k_j$
  Find $t \in N_0$ such that $\frac{c^i_t}{w_t} = \max\left\{\frac{c^i_j}{w_j} : w_j \leqslant \bar{w}, j \in N_0\right\}$
  IF (t exists) THEN
  BEGIN
    $x \leftarrow x^k$;
    $\overline{z_1} \leftarrow \sum_{j=1}^{n} c^1_j x_j; \overline{z_2} \leftarrow \sum_{j=1}^{n} c^2_j x_j$
    WHILE (t exists) DO
    BEGIN
      $\overline{z_1} \leftarrow \overline{z_1} + c^1_t; \overline{z_2} \leftarrow \overline{z_2} + c^2_t$; $\bar{w} \leftarrow \bar{w} - w_t$; $x_t \leftarrow 1$
      $N_0 \leftarrow N_0 \setminus \{t\}$
      Find $t \in N_0$ such that $\frac{c^i_t}{w_t} = \max\left\{\frac{c^i_j}{w_j} : w_j \leqslant \bar{w}, j \in N_0\right\}$
    END
    Update $\widetilde{X}$ with the x
  END
END

In the second case, the inclusion of the item corresponding to the fractional variable is only possible if at least one of the included items is removed. Two new solutions are created using two heuristic procedures. The first (second) corresponds to the remotion of the item which makes the solution feasible and has the lowest profit-to-weight ratio, according to criterion z_1 (z_2). The process of including an item f from solution x^k and removing another (taking into account both criteria) is called negative bidimensional improvement of the solution. The specific steps are given below.

Negative bidimensional improvement (x^k, f)

FOR $i = 1$ **TO** 2 **DO** {criteria number}
BEGIN
$N_1 = \{j : x_j^k = 1, j = 1, \ldots, n\} \setminus \{f\}$
$x_f^k \leftarrow 1$
$\bar{w} \leftarrow W - \sum_{j \in N_1} w_j - w_f$
Find $t \in N_1$ such that $\frac{c_t^i}{w_t} = \min\left\{\frac{c_j^i}{w_j} : w_j \geqslant -\bar{w}, j \in N_1\right\}$
IF (t exists) **THEN**
BEGIN
$x \leftarrow x^k$;
$N_0 = \{j : x_j = 0, j = 1, \ldots, n\}$
$x_t \leftarrow 0$
$\overline{z_1} \leftarrow \sum_{j \in N_1} c_j^1 + c_f^1 - c_t^1$; $\overline{z_2} \leftarrow \sum_{j \in N_1} c_j^2 + c_f^2 - c_t^2$; $\bar{w} \leftarrow \bar{w} + w_t$
Find $t \in N_0$ such that $\frac{c_t^i}{w_t} = \max\left\{\frac{c_j^i}{w_j} : w_j \leqslant \bar{w}, j \in N_0\right\}$
WHILE (t exists) **DO**
BEGIN
$\overline{z_1} \leftarrow \overline{z_1} + c_t^1$; $\overline{z_2} \leftarrow \overline{z_2} + c_t^2$
$\bar{w} \leftarrow \bar{w} - w_t$
$x_t \leftarrow 1$
$N_0 \leftarrow N_0 \setminus \{t\}$
Find $t \in N_0$ such that $\frac{c_t^i}{w_t} = \max\left\{\frac{c_j^i}{w_j} : w_j \leqslant \bar{w}, j \in N_0\right\}$
END
Update $\widetilde{X}$ with the x
END
END

Notice that, when a solution from the diversification method is already feasible for problem (1), the knapsack is full. In this case, it is only possible to consider set to 0 the basic variable when its value is equal to 1. This is carried out as described above.

The while loop, in both the positive and negative bidimensional improvement procedures, is the well known greedy procedure applied to the single criterion reduced knapsack problem:

$$\max\left\{\sum_{j \in N_0} c_j^i x_j : \sum_{j \in N_0} w_j x_j \leqslant \bar{w}, x_j \in \{0,1\}, j \in N_0\right\}, \quad i = 1, 2. \tag{5}$$

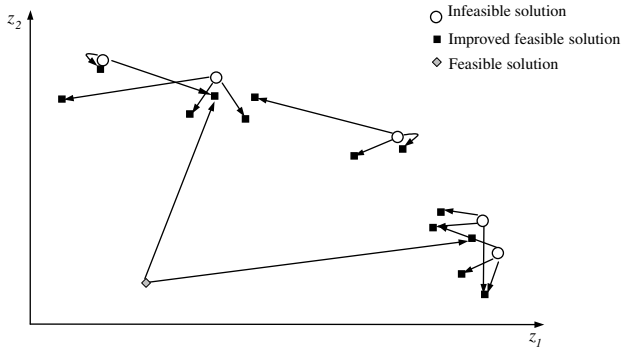

Fig. 1. Improvement method.

In the combination method (Section 3.5) the solutions are combined in such a way that feasibility is always maintained. The improvement of these solutions involves using the positive and negative bidimensional improvement procedures with one variable taking the role of the fractional variable.

The solutions obtained by the heuristics are considered to be included in $\widetilde{X}$, which is empty at the beginning of the method. If a solution is not dominated by any of the members of $\widetilde{X}$, then it is included and $\widetilde{X}$ is updated by removing the solutions that are dominated by the inserted one. If not, it is discarded.

Fig. 1 shows the behavior of the improvement method applied to both infeasible and feasible solutions. In this figure, the arrows link an improved solution to the solutions obtained with the improvement method. From infeasible solutions (generated in the diversification method), the improvement method explores neighborhoods that are very close to them (because the feasible region is very limited). The residual capacity is small, and thus only a few items are considered to generate improved solutions. On the other hand, when starting from a feasible solution it is possible to achieve solutions situated in far regions which are not obtained by exploring the nearest neighbourhood of infeasible solutions. This is due to the fact that a larger set of items can now be considered.

3.3. The reference set update method

Generally, the reference set is constructed by considering all the different solutions previously obtained, but here we only consider the list of potentially efficient solutions, $\widetilde{X}$, invoking a kind of "elitism". Thus the construction of reference set, R, is based on the concept of efficient solution. However, in large size instances, $\widetilde{X}$, can be huge just after only a few iterations, making it necessary to use a kind of clustering technique. A subset of the reference set is usually composed of the *k-best solutions*. Nevertheless, in multiple criteria problems we cannot distinguish, *a priori*, between efficient solutions.

Property 2 can be used to establish the way in which the reference set is built. According to this property, only a small number of variables of a given solution need to be changed in order to obtain other PNDS. If very close solutions (in the decision space) are considered in the reference set, it is possible that after the application of Property 2 a small proportion of new solutions will be found. In order to avoid this, the solutions of $\widetilde{X}$ are ordered according to non-decreasing values of the following dissimilarity measure:

$$d_k^{\text{ref}} = \|x^{\text{ref}} - x^k\| = \sum_{j=1}^{n} |x_j^{\text{ref}} - x_j^k|, \quad k = 1, \ldots, |\widetilde{X}|, \tag{6}$$

where x^{ref} is the solution with the highest value on z_2 (it could also be z_1), and R is built from $\widetilde{X}$, by dividing it into clusters, with approximately the same number of solutions. One solution (the mean solution), from each cluster, is selected in order to define the reference set, R. We consider a maximum of 20 solutions, and from these 20 clusters are thus built.

3.4. The subset generation method

The subsets of the reference set define the solutions to be combined in order to create new ones. According to Property 2 we should explore a neighborhood of an efficient solution (in our case, potentially efficient) aiming to get new ones. As the combination method is based on the exploration of the distinctions between solutions in each subset, we should not consider very dissimilar solutions from the reference set. In this way, we will try the following very simple strategy: the subsets will be considered to be pairs of consecutive solutions from the reference set. Hence, there will be $|R-1|$ subsets to be considered.

As there is no point in examining the same subset several times, a *Tabu-list* is created, and each subset is included in it as long as that subset is not already part of the list. In this case it is discarded. Thus, the *Tabu-list* is a record of the already examined subsets. But, due to expensive memory requirements to maintain vectors with n bits, we opted for saving only their image in the criteria space, incurring the risk of losing some alternative solutions, i.e., different solutions in the decision space with the same image in the criteria space.

3.5. The solution combination method

Based on Property 2 we only have to change the value of a small number of items in order to obtain other potentially efficient solutions. We used a combination method that explores this property by creating a path between the solutions of each subset. Let x^0 and x^1 be the solutions of one subset; in the nomenclature of Glover (1999) x^0 is called the *initiating* solution and x^1 is called the *guiding* solution. The *guiding* solution is used to identify the variables whose values should change in the initiating solution. This means that several new solutions are created from x^0 with the property of incorporating characteristics of x^1. This is achieved by detecting the variables whose values differ in x^0 and x^1, i.e., the items which are to be removed from x^0 and the items which are to be inserted into x^0. Removing/inserting these items one by one, we then search for a positive and negative bidimensional improvement of the solution on the decision space confined to the variables equal to 0/1 in x^0. The exploration of the decision space is highly conditioned due to the low residual capacity that comes from removing/inserting only just one item. Only a small neighborhood around x^0 is considered, and this will suffice according to Property 2. The combination method can preserve a common structure corresponding to the position where the variables in the two solutions have the same value. However, as these variables can also be used to obtain enhanced solutions, it is possible to achieve others with new distinguishing features different from x^1.

Since the path from x^0 to x^1 is different from the path x^1 to x^0, the solutions are swapped and once again the above combination method is applied. In this way, new regions of the decision space will be searched, enabling the creation of new solutions. A description of the combination method is given below:

Each time a variable, j, from F_1^0 is set to 0 the feasible region is restricted to $\sum_{i:x_i^0=0} w_i x_i \leqslant W - \sum_{i:x_i^0=1} w_i + w_j$, and as $W - \sum_{i:x_i^0=1} w_i + w_j$ is small, only a few items need to be considered. The same thing occurs when a variable from F_1^0 is set to 1.

Combination method (x^0, x^1)

$F_1^0 \leftarrow \{j : x_j^0 \neq x_j^1, x_j^0 = 1\}$ {items to be removed}
$F_0^0 \leftarrow \{j : x_j^0 \neq x_j^1, x_j^0 = 0\}$ {items to be inserted}
$j = 1$
WHILE $j \leqslant |F_1^0|$ **DO**
BEGIN
  $f_j \leftarrow$ {element j of F_1^0}
  Positive bidimensional improvement (x^0, f_j)
  $j = j + 1$

END
$j = 1$
WHILE $j \leqslant |F_0^0|$ **DO**
BEGIN
  $f_j \leftarrow$ {element j of F_0^0}
  Negative bidimensional improvement (x^0, f_j)
  $j = j + 1$
END

In the positive and negative bidimensional improvement procedures, the search benefits from the above mentioned order of items: $w_1 \leqslant \cdots \leqslant w_n$. If one item does not fit into the knapsack, the forward items do not need to be considered.

In Fig. 2 the result of the combination method applied to (x^1, x^2) and (x^2, x^3) is depicted. The solutions represented by squares were obtained by combining the solutions represented by circles. The dashed arrows correspond to the direct path between the solutions, and the full arrows correspond to the reverse path. The direction of the arrows does not have any specific significance. These arrows are only used to link the combined solutions. In the decision space, the distance between x^2 and x^3 is greater than the distance between x^1 and x^2. This enables the discovery of more diversified solutions, since more items are considered to change

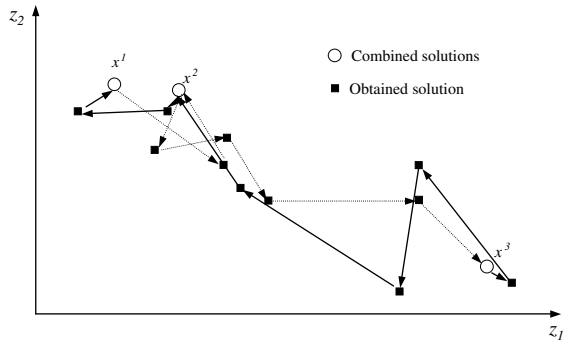

Fig. 2. Combination method.

their values. By swapping the roles of initiating and guiding solutions different solutions, not pertaining to the first paths, were obtained: (x^1, x^2) and (x^2, x^3).

3.6. Overall description of the scatter search method

$\widetilde{X} = \phi$; *Tabu_list* $= \phi$
//Generate an initial trial solutions set
Let x^0 be the solution that maximizes criterion z_2 in the relaxed problem, and f^0 be the index of the basic variable
Apply the **Positive** and **Negative Bidimensional Improvement** with (x^0, f^0) as inputs
$k = 0$
WHILE (x^k does not optimize z_1) **DO**
BEGIN
Generate the efficient solution adjacent to x^k. Let x^{k+1} be such a solution, and f^{k+1} the index of the basic variable in x^{k+1}
Apply the **Positive** and **Negative Bidimensional Improvement** with (x^{k+1}, f^{k+1}) as inputs
$k = k + 1$

END
//Create new solutions
Terminate = false
WHILE (Terminate = false)**DO**
BEGIN
Apply the **Reference Set Update Method** and let R small be the reference set obtained
Apply the **Subset Generation Method** and let *nsub* be the number of created subsets
$k = 1$
WHILE ($k \leqslant nsub$) **DO**
BEGIN
Let x^0 and x^1 be the solutions of subset k
IF ($z(x^0)$, $z(x^1)$) $\notin$ *Tabu-List* **THEN**
BEGIN
Apply the **Combination Method** (x^0, x^1)
Insert ($z(x^0)$, $z(x^1)$) in the *Tabu-List*
//Swap solutions
Apply the **Combination Method** (x^1, x^0)
Insert ($z(x^1)$, $z(x^0)$) in the *Tabu-List*
END
$k = k + 1$
END
IF (stopping condition verified) then Terminate = true
END

The most frequent stopping condition is the maximum number of iterations. The method is repeated until this number is achieved. However, one important factor in evolutionary algorithms is the time required. Unfortunately, we do not have, *a priori*, a good estimate of the computational time consumption involved for each iteration. Fixing the number of iterations could involve too much time or cause

premature stopping. This is especially because in general the more time is taken the more solutions are found. Another important input, that could be used as a stopping condition, is the desired quality of the solutions so far obtained.

In the computational experiments we will use a controlled number of iterations as the stopping condition.

4. Computational experiments and results

In approximate methods, the quality of solutions is a topic of extreme relevance. Two key issues which are used to evaluate the quality of an approximation are the proximity to the set of exact non-dominated solutions and the diversity of those solutions (Zitzler, 1999a; Deb, 2001; Coello et al., 2002). For the first issue we used the relaxed problem (2) to give a measure of the maximum error implicit in every solution. This was because it is not possible to generate the set of exact solutions for most of the considered instances. From Section 2, an upper frontier for the non-dominated solutions of (1) was derived. For each of the non-dominated solution, $z^+ = (z_1^+, z_2^+)$, the nearest point in the upper frontier, $z^* = (z_1^*, z_2^*)$, was calculated according to the L_∞ metric. These two points are then used to derive the gradient of a weighted sum function: $f(z) = \pi_1 z_1 + \pi_2 z_2$, with $z = (z_1, z_2)$, $\pi_1 = z_1^* - z_1^+$ and $\pi_2 = z_2^* - z_2^+$. The percentage gap between points z^+ and z^* was calculated using $f(z)$:

$$\frac{f(z^*) - f(z^+)}{f(z^*)} \times 100. \tag{7}$$

Two measures were considered to evaluate the diversity of the solutions in the criteria space. The first, is the standard deviation of the Euclidean distances between consecutive PNDS in the criteria space, and is similar to that proposed by Schott (1995).

The second, is based once again on the derived upper frontier, which is used to divide the criteria space into regions. The expected number of solutions *per* region was computed and the standard deviation of the number of solutions *per* region was considered. The lower this value is the greater is the dispersion of the obtained solutions along the upper frontier.

The computational experiments concern large size instances where $n = 1000$, $n = 2000$ up to $n = 6000$. The coefficients are integer numbers randomly and uniformly generated within the range [1, 1000] and the knapsack capacity is 50% of the sum of the weights. For each value of n, 15 instances were generated. The computational experiments were performed on a Pentium 4 processor at 1495 MHz with 256 MB RAM and 40 GB hard disk. The method was implemented in Borland Delphi 4.

The stopping condition was considered to be the limit of 15 iterations after the diversification method had been applied, and the subset cardinality was fixed at 20 solutions.

In Table 3, the results concerning all the 90 instances are given. Column 1 refers to the number of items; column 2 concerns the computational time in seconds; column 3 shows the number of potentially non-dominated solutions; columns 4–7 refer to the L_∞ metric; column 8 presents the standard deviation of the number of solutions *per* region, and column 9 is the standard deviation of the Euclidean distance between consecutive PNDS.

Results show that for all the instances considered the approximation to the upper frontier is quite good. The percentage distance (column 4, Table 3) is insignificant, and the diversity of these values is very small, as can be seen by its small standard deviation and amplitude (columns 5–7, Table 3). The solutions are well dispersed along the criteria space as the standard deviation of the solutions by region and the distance between consecutive solutions, are both small (columns 8 and 9, Table 3).

Table 3
Overall results

	L-infinite distance (%)							Diversity	
	(1)	(2)	(3)	(4)	(5)	(4)	(7)	(8)	(9)
	n	CPU (s)	PNDS	Average	Max	Min	STD	STD-C	STD-D
Average	1000	58.85	1552.60	0.0137	0.0474	0.0002	0.0001	5.91	121.56
Max		65.90	1694.00	0.0146	0.0744	0.0008	0.0001	6.39	137.86
Min		51.25	1398.00	0.0127	0.0360	0.0000	0.0001	5.21	106.99
STD		3.92	78.69	0.0006	0.0100	0.0002	0.0000	0.36	8.35
Average	2000	239.21	2958.40	0.0061	0.0264	0.0001	0.0000	8.25	133.24
Max		263.25	3223.00	0.0065	0.0359	0.0002	0.0000	9.21	144.90
Min		222.40	2732.00	0.0056	0.0210	0.0000	0.0000	7.28	116.60
STD		12.33	154.40	0.0002	0.0039	0.0001	0.0000	0.53	8.53
Average	3000	486.96	4150.80	0.0037	0.0180	0.0001	0.0000	9.48	158.67
Max		579.74	4614.00	0.0040	0.0229	0.0001	0.0000	9.99	413.70
Min		423.70	3819.00	0.0035	0.0138	0.0000	0.0000	8.38	132.80
STD		36.41	228.84	0.0002	0.0026	0.0000	0.0000	0.43	68.33
Average	4000	975.45	5283.20	0.0026	0.0156	0.0000	0.0000	11.11	147.18
Max		1121.74	5738.00	0.0027	0.0323	0.0000	0.0000	12.35	157.09
Min		834.31	4641.00	0.0024	0.0109	0.0000	0.0000	9.92	138.36
STD		79.70	295.97	0.0001	0.0054	0.0000	0.0000	0.58	4.93
Average	5000	1545.96	6750.60	0.0020	0.0111	0.0000	0.0000	12.66	144.96
Max		1762.10	7520.00	0.0021	0.0135	0.0001	0.0000	14.50	159.40
Min		1345.10	5888.00	0.0018	0.0084	0.0000	0.0000	11.37	131.57
STD		94.70	422.71	0.0001	0.0013	0.0000	0.0000	0.86	7.59
Average	6000	2263.61	8200.20	0.0016	0.0096	0.0000	0.0000	13.89	144.63
Max		2505.70	9178.00	0.0017	0.0196	0.0000	0.0000	15.30	168.14
Min		1801.61	6851.00	0.0014	0.0068	0.0000	0.0000	12.26	137.02
STD		182.71	597.33	0.0001	0.0030	0.0000	0.0000	0.72	7.47

The computational time is reasonable for the size of the instances, although it displays nonlinear behavior with that size. In fact, Fig. 3 shows that the CPU time, on average, is a power of the number of items.

Another interesting aspect to be observed is how the set of the PNDS along the iterations evolves and how the computational time is spent. Let us consider the instance with the highest number of potentially

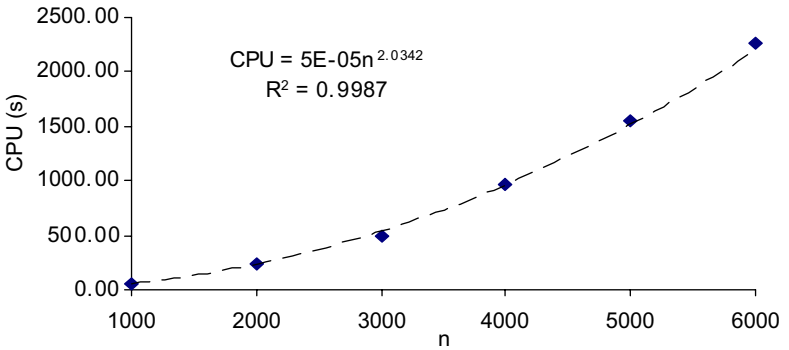

Fig. 3. Average CPU time by instance size.

non-dominated solutions as the reference for each size. In Fig. 4 the number of PNDS increases quickly in the first iterations, but then more slowly. As can be seen in Fig. 5, the increasing rate of the number of PNDS converges to zero within a small number of iterations. This means that the method could have been stopped earlier without a substantial reduction of the PNDS set. Notice, however, that the difference in PNDS between an iteration and the one previous to it, is less than the number of PNDS obtained in the current iteration, as the set $\widetilde{X}$ is "cleaned" of the solutions which were dominated by the current solutions. This is also due to the generation of high quality solutions.

The computational time shows linear behavior with the number of iterations (Fig. 6) and the time *per* iteration is very regular (Fig. 7). Especially after the first few iterations. This is related to the values shown in Fig. 5.

The solutions obtained from the diversification method, with their consequent improvement, appear to be very good as more than 75% of them (column 3 in Table 4, which is obtained by dividing values of column 2 by values of column 1) remain in the final set of PNDS. Taking into account the time required in the diversification method, which is less than 6% of the total time spent, and considering that the solutions obtained represent more than 30% (column 5, in Table 4, which is obtained by dividing values of column

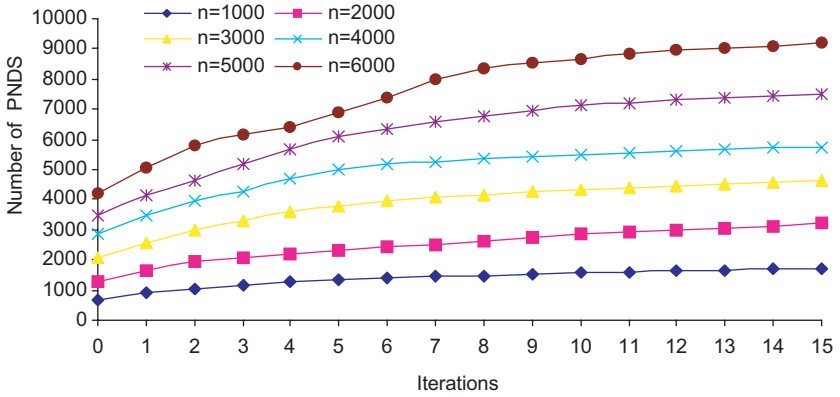

Fig. 4. PNDS per iteration.

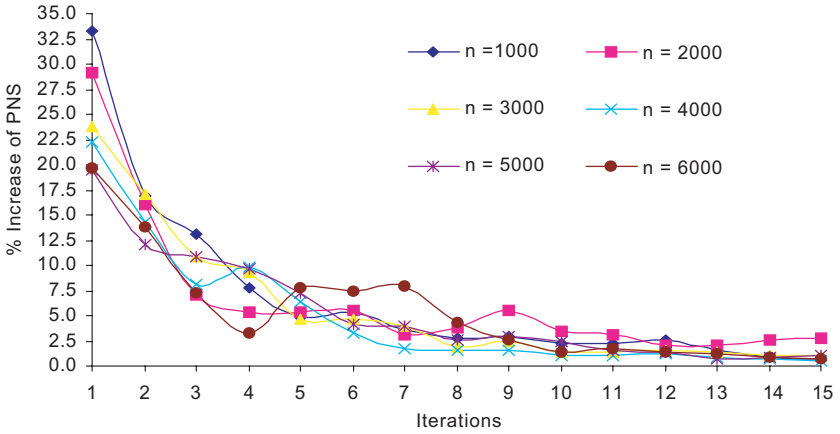

Fig. 5. % Increase of PNDS.

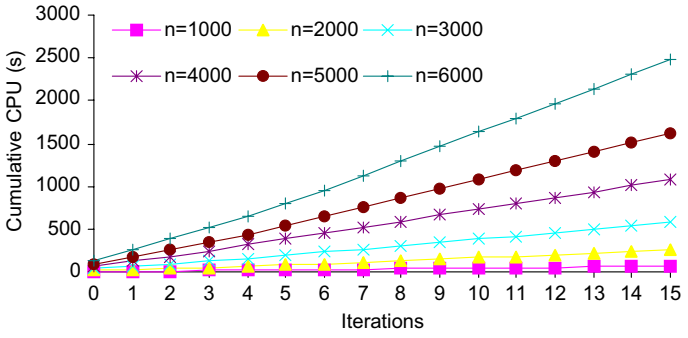

Fig. 6. Cumulative CPU time.

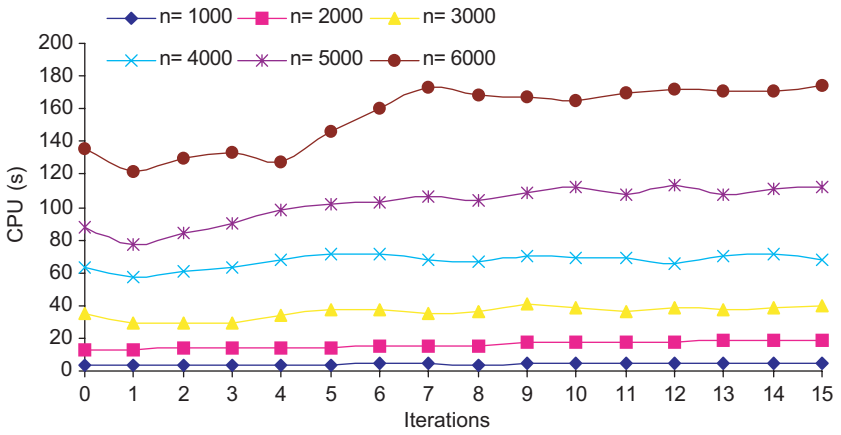

Fig. 7. CPU time per iteration.

Table 4
Influence of the initial set

n	(1)	(2)	(3)	(4)	(5)
1000	686	515	75.1	1694	30.4
2000	1298	1005	77.4	3223	31.2
3000	2048	1694	82.7	4614	36.7
4000	2838	2428	85.6	5738	42.3
5000	3478	3041	87.4	7520	40.4
6000	4235	3783	89.3	9178	41.2

(1)—PNDS in the initial set.
(2)—Solutions from the initial set in the final set.
(3)—percentage of solutions from the initial set that survived through the iterations.
(4)—PNDS in the final set.
(5)—percentage of solutions from the initial set in the final set.

2 by values of column 4) of the total number in the final PNDS set, one can say that the diversification method is the most efficient component of the proposed SS method.

Although the results seem very good in all aspects considered, it must be pointed out that the number of solutions is less than the number of exact non-dominated solutions obtained by Visée et al. (1998) and Captivo et al. (2003). In their research the number of solutions appears to have an exponential growth with the number of items, which is contrary to our results, where behavior tends to be linear (Fig. 8).

In Fig. 9 the PNDS set of an instance with 3000 items is shown. The figure on the right shows an amplified section taken from that on the left, and expresses the upper frontier and the initial solutions, both derived in the diversification method (Fig. 9, top right) and the result of 15 iterations of the SS proposed method (Fig. 9, bottom right). As can be seen, a great improvement was made concerning the exploration of the criteria space.

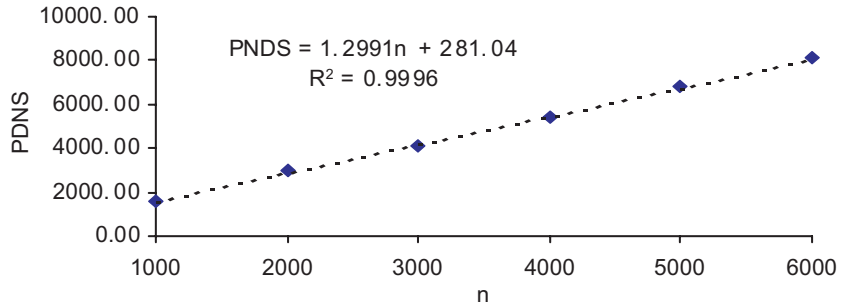

Fig. 8. Average PNDS by instance size.

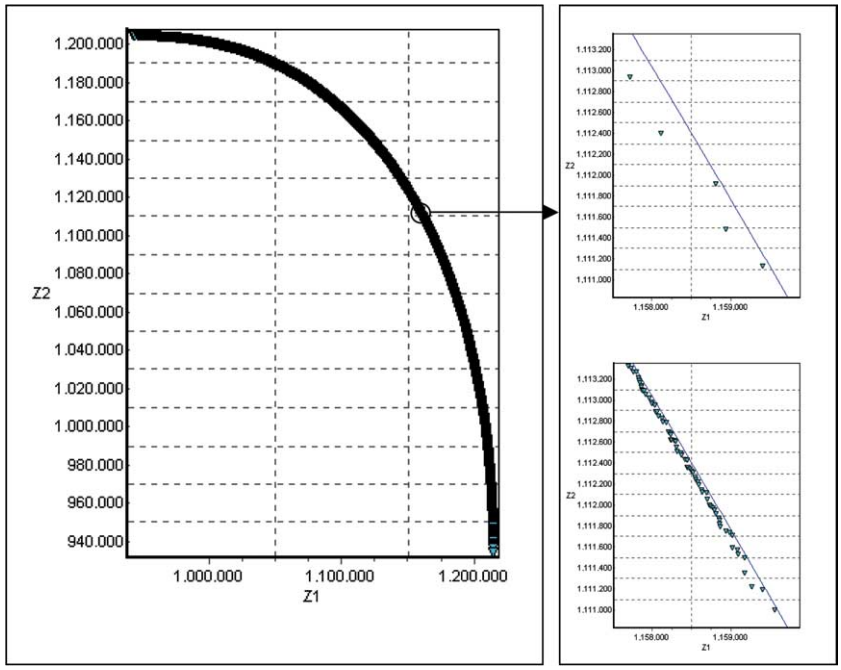

Fig. 9. PNDS set.

4.1. Comparison with exact methods

This section presents the comparison between the set of solutions obtained with the SS method (PNDS) and the exact set of non-dominated solutions (NDS) for 15 instances of the problem with $n = 100$, 300, 500. The NDS was determined using an implementation of the exact method by Visée et al. (1998).

The comparison concerns: (1) the number of solutions, (2) the computational time, (3) the percentage of exact non-dominated solutions, and (4) the gap between dominated and non-dominated solutions. The gap is computed as follows. Consider a solution from PNDS, $(\tilde{z}_1^k, \tilde{z}_2^k)$, which is dominated by a solution from NDS, (z_1^t, z_2^t). The gap between $(\tilde{z}_1^k, \tilde{z}_2^k)$ and (z_1^t, z_2^t) is $|z_1^t - \tilde{z}_1^k|/z_1^t$ if $|z_1^t - \tilde{z}_1^k| > |z_2^t - \tilde{z}_2^k|$ or $|z_2^t - \tilde{z}_2^k|/z_2^t$ otherwise. All the solutions from NDS are tested, and the minimum gap is considered to be the gap between $(\tilde{z}_1^k, \tilde{z}_2^k)$ and the NDS set. This is repeated for all the dominated solutions from PNDS, and the gap between PNDS and NDS is set equal to the maximum of the computed gaps.

All the parameters assumed in the experiments for large size instances were kept the same. Thus, the reference set is composed of 20 solutions, and 15 iterations were set as the stopping condition. In Table 5 it is presented the number of exact non-dominated solutions and the potentially non-dominated solutions, the CPU in seconds for computing each of the sets. The relation between these two attributes are also presented. As can be observed, the number of solutions in NDS is greater than in PNDS and, on average, is 1.53 times the number of PNDS when $n = 100$; 2.14 times when $n = 300$ and 2.45 times when $n = 500$. The CPU time is increasingly favourable for the SS method. When $n = 500$, the exact method takes 408.73 times the computational time required by the SS method.

Table 5
CPU time and number of exact and potentially non-dominated solutions

n	Statistic	Exact method		SS method		(1)*	(2)*
		NDS	CPU(s)_1	PNDS	CPU(s)_2		
100	Average	125.47	4.14	84.47	0.33	1.53	12.54
	Max	176.00	7.03	130.00	0.49	1.92	16.50
	Min	73.00	2.20	49.00	0.16	0.57	9.18
	STD	29.14	1.16	21.28	0.08	0.30	2.21
300	Average	792.93	507.27	371.67	4.20	2.14	120.39
	Max	917.00	807.90	452.00	5.39	2.37	169.65
	Min	612.00	306.32	272.00	3.03	2.02	76.44
	STD	86.87	183.95	49.08	0.68	0.10	27.30
500	Average	1588.40	5031.43	644.87	12.30	2.45	408.73
	Max	1830.00	7706.54	730.00	14.83	2.72	661.86
	Min	1348.00	3600.36	563.00	10.60	2.25	308.16
	STD	144.15	1254.07	48.30	1.33	0.12	94.68

* (1) NDS/PNDS; (2) CPU(s)_1/CPU(s)_2.

Table 6
Quality of PNDS

	$n = 100$				$n = 300$				$n = 500$			
	Av.	Max	Min	STD	Av.	Max	Min	STD	Av.	Max	Min	STD
(1)	33.13	52.05	25.57	6.64	9.75	14.36	7.34	1.83	5.12	7.48	3.53	1.06
(2)	5.87	16.00	4.00	2.94	3.73	8.00	2.00	1.65	2.73	4.00	2.00	0.68

(1)—percentage of exact non-dominated solutions.
(2)—gap × 1000.

The structure of the SS enables the determination of good solutions within a competitive computational time, as revealed by the small gap between the dominated solutions of PNDS and solutions from NDS (line (2) in Table 6). On average, the maximum gap varies from 5.87/1000 and 2.73/1000 for $n = 100$ to $n = 500$. However, it must be pointed out that the SS method fails in obtaining exact non-dominated solutions. The percentage of exact non-dominated solutions in PNDS substantially decreases with the problem size. In our experiments, it is about 33% for $n = 100$ but only 5% for $n = 500$ (line (1) in Table 6).

5. Conclusions and future research

We applied scatter search methodology for generating an approximation of the set of non-dominated solutions for large size bi-criteria knapsack instances. Properties of the single criterion and bi-criteria instances were used to guide and generate solutions. The results showed that the time required to get an accurate description of the set of non-dominated solutions is relatively small even for large size instances, which is a very important feature of the proposed method. This rapidly obtained information can be used as a powerful tool in an interactive context, giving the decision maker a good estimate of the non-dominated solutions. This enables him to localize and focus his attention on interesting areas, that can be further explored by exact methods. However, it has been verified that the method was not able to find the complete set of non-dominated solutions for the large size instances considered. Developing more intensification strategies maybe required, but this will increase computational time. Consequently, a compromise between these two objectives must always be established. The method used a rigid structure for generating subsets and for combining solutions. It is possible that the stagnation of the number of PNDS reached after just a few iterations is due to this feature. Thus, making these structures more flexible will probably enhance the intensification requirements.

The approach here presented can be easily extended to more than two criteria. However, the extension of the diversification method is the most problematic component once the detection of all the directions to obtain extreme non-dominated solutions of the linear relaxation should be performed. Although, it should be noticed that special care must be taken to keep computational time down. So, a less demanding version could be considered, i.e., generate a subset of well dispersed directions to search for some extreme non-dominated solutions. The rest of the components of the method are trivially extended once they only involve the search along single criterion directions.

Acknowledgments

The authors would like to acknowledge the support from MONET research project (POCTI/GES/37707/2001) and the third author also acknowledges the support of the grant SFRH/BDP/6800/2001 (Fundação para a Ciência e Tecnologia, Portugal).

References

Balicki, J., Kitowski, Z., 2001. Multicriteria evolutionary algorithm with tabu search for task assignment. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D. (Eds.), Proceedings of the First International Conference on Evolutionary Multicriterion Optimization, Lecture Notes in Computer Science, vol. 1993. Springer, Berlin, pp. 373–384.

Bazaraa, M., Jarvis, J., Sherali, H, 1992. Linear Programming and Network Flows, second ed. John Wiley, New York.

Beausoleil, R.P., 2001. Multiple criteria scatter search. In: Proceedings of the 4th Metaheuristics International Congress, Porto, Portugal, pp. 539–543.

Ben Abdelaziz, F., Krichen, S., Chaouachi, J., 1999. A hybrid heuristic for multiobjective knapsack problems. In: Voss, S., Martello, I., Osman, I., Roucairol, C. (Eds.), Meta-heuristics: Advances an Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers, Dordrecht, pp. 205–212.

Captivo, M., Clímaco, J., Figueira, J., Martins, E., Santos, J.L., 2003. Solving multiple criteria 0–1 knapsack problems using a labeling algorithm. Computers & Operations Research 30, 1865–1886.

Corberán, A., Fernández, E., Laguna, M., Martí, R., 2002. Heuristic solutions to the problem of routing school buses with multiple objectives. Journal of the Operational Research Society 53 (4), 427–435.

Coello, C., Veldhuizen, D., Lamont, G., 2002. Evolutionary Algorithms for Solving Multi-objective Problems. Kluwer Academic Publishers, Dordrecht.

Czyzak, P, Jaskiewicz, A., 1998. Pareto simulated annealing—A metaheuristic technique for multiple objective combinatorial optimization. Journal of Multicriteria Decision Analysis 7, 34–47.

Deb, K., Goel, T., 2001. A hybrid multi-objective evolutionary approach to engineering shape design. In: Zitzler, E., Deb, K., Thiele, C.A., Coello Coello, C.A., Corne, D. (Eds.), Proceedings of the First International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, vol. 1993. Springer, Berlin, pp. 385–399.

Deb, K., 2001. Multi-objective Optimization using Evolutionary Algorithms. John Wiley, New York.

Gandibleux, X., Fréville, A., 2000. Tabu search based procedure for solving the 0–1 multiobjective knapsack problem: The two objectives case. Journal of Heuristics 6, 361–383.

Gandibleux, X., Morita, H., Katoh, N., 2001. The supported solutions used as a genetic information in a population heuristic. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D. (Eds.), Proceedings of the First International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, vol. 1993. Springer, Berlin, pp. 429–442.

Glover, F., 1999. Scatter search and path relinking. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas in Optimization. McGraw-Hill, pp. 297–316.

Ishibuchi, H., Murata, T., 1998. Multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews 28 (3), 392–403.

Jaszkiewicz, A., 2001. Comparison of local search-based metaheuristics on the multiple objective knapsack problem. Foundations of Computing and Decision Sciences 26 (1), 99–120.

Jaszkiewicz, A., 2002a. On the performance of multiple objective genetic local search on the 0/1 knapsack problem. A comparative experiment. IEEE Transactions on Evolutionary Computation 6 (4), 402–412.

Jaszkiewicz, A., 2002b. Genetic local search for multiple objective combinatorial optimization. European Journal of Operational Research 137 (1), 50–71.

Jaszkiewicz, A., 2003. Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem. IEEE Transactions on Evolutionary Computation 7 (2), 133–143.

Knowles, J.D., Corne, D.W., 2000. M-PAES: A memetic algorithm for multiobjective optimizationProceedings of the 2000 Congress on Evolutionary Computation, vol. 1. IEEE Service Center, Piscataway, NJ, pp. 325–332.

Martello, S., Toth, P., 1990. Knapsack Problems—Algorithms and Computer Implementations. John Wiley, New York.

Martí, R., Laguna, M., Lourenço, H., 2000. Assigning proctors to exams with scatter search. In: Laguna, M., González-Velarde, J.L. (Eds.), Computing Tools for Modelling, Optimization and Simulation: Interfaces in Computer Science and Operations Research. Kluwer Academic Publishers, Boston, pp. 215–227.

Morita, H., Gandlibleux, X., Katoh, N., 2001. Experimental feedback on biobjective permutation problems solved with a population heuristic. Foundations of Computing and Decision Sciences 26 (1), 23–50.

Murata, T., Ishibuchi, H., Gen, M., 2000. Cellular genetic local search for multi-objective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference 2000, pp. 307–314.

Pisinger, D., Toth, P., 1998. Knapsack problems. In: Du, D.-Z., Pardalos, P. (Eds.), Handbook of Combinatorial Optimization, vol. 1. Kluwer Academic Publishers, Dordrecht.

Schott, J.R., 1995. Fault tolerance design using single and multi-criteria genetic algorithms. Master's thesis, Massachusetts Institute of Technology.

Steuer, R., 1986. Multiple Criteria Optimization, Theory, Computation and Application. John Wiley, New York.

Talbi, E., Rahoual, M., Mabed, M., Dhaenens, C., 2001. A hybrid multi-objective evolutionary approach to engineering shape design. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D. (Eds.), Proceedings of the First International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, vol. 1993. Springer, Berlin, pp. 416–428.

Visée, M., Teghem, J., Ulungu, E.L., 1998. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. Journal of Global Optimization 12, 139–155.

Zitzler, E., 1999a. Evolutionary algorithms for multiobjective optimization: Methods and applications. Ph.D. dissertation, Swiss Federal Institute of Technology of Zurich, Zurich.

Zitzler, E., Thiele, L., 1999b. Multiple objective Evolutionary Algorithms: A comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3 (4), 257–271.