Journal of Mathematical Modelling and Algorithms **3:** 183–208, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

A Scatter Search Method for the Bi-Criteria Multi-dimensional {0,1}-Knapsack Problem using Surrogate Relaxation

CARLOS GOMES DA SILVA[1,★], JOÃO CLÍMACO[2] and JOSÉ FIGUEIRA[2,★★]
[1] *INESC-Coimbra and Escola Superior de Tecnologia e Gestão de Leiria, Morro do Lena, Alto Vieiro, 2401-951 Leiria, Portugal. e-mail: cgsilva@estg.iplei.pt*
[2] *INESC-Coimbra and Faculdade de Economia, Universidade de Coimbra, Av. Dias da Silva, 165, 3004-512 Coimbra, Portugal. e-mail: jclimaco@inescc.pt, figueira@fe.uc.pt*

Abstract. This paper presents a scatter search (SS) based method for the bi-criteria multi-dimensional knapsack problem. The method is organized according to the usual structure of SS: (1) diversification, (2) improvement, (3) reference set update, (4) subset generation, and (5) solution combination. Surrogate relaxation is used to convert the multi-constraint problem into a single constraint one, which is used in the diversification method and to evaluate the quality of the solutions. The definition of the appropriate surrogate multiplier vector is also discussed. Tests on several sets of large size instances show that the results are of high quality and an accurate description of the entire set of the non-dominated solutions can be obtained within reasonable computational time. Comparisons with other meta-heuristics are also presented. In the tested instances the obtained set of potentially non-dominated solutions dominates the set found with those meta-heuristics.

Mathematics Subject Classification (2000): 90C27.

Key words: meta-heuristics, multi-dimensional knapsack problem, scatter search method, surrogate relaxation, combinatorial optimization.

1. Introduction

The multi-dimensional {0,1}-knapsack problem (MDKP) has a combinatorial nature aiming to maximize the sum of the values of the items to be selected from a given set, taking into account several resource constraints. It is an extension of the well-known {0,1}-knapsack problem. The only difference concerns the number of the resource constraints. In the latter this number is only one and in the former it is more than one.

The MDKP has many applications in the fields of capital budgeting, cutting stock, cargo loading, allocating processors and databases in distributed computer systems [8]. Due to its hard complexity (it is an NP-complete combinatorial prob-

★ Corresponding author. This email is available for all problems and questions.
★★ Visiting Researcher (August 2002–May 2003) at DIMACS Center, Rutgers University, 96 Frelinghusysen Road, Piscataway, NJ 108855-8018, USA.

lem) the size of the instances which can be solved exactly is modest. Frévillle and Plateau [6] mentioned the limit of 5 constraints and 200 variables for previous works. Exact methods are based on dynamic programming [9, 29] or in branch-and-bound techniques [26, 8, 23]. Large size instances can only be managed with approximate methods which establish a balance between the quality of the solutions and the computational time required. And, in fact, the literature is much richer in these kind of approaches [20, 22, 3, 15, 1, 19, 28]. An extensive survey of the research developed in the MDKP can be found in Chu and Beasley [3], where the works are classified into exact and heuristic algorithms.

Many practical situations require the incorporation of several conflicting criteria. In general, in a multiple criteria model there is no a feasible solution that optimizes simultaneously all the criteria. Consequently a compromise among the criteria must be achieved. Solving these models consists of determining compromise solutions, called *non-dominated/efficient solutions*. One of the most popular approaches to "solve" multiple criteria problems aims to generate the whole set of the non-dominated/efficient solutions [18]. Nevertheless, like in the single criterion instances, computational requirements involved in large size ones, forbid the determination of that set. Thus, approximate methods, as meta-heuristics, were developed for generating an approximation of the non-dominated solutions set. Actually the solutions may be dominated, and for this reason, they are called *potentially non-dominated solutions*. Several meta-heuristics were designed to tackle multiple criteria combinatorial problems. Nevertheless, they are considerably less than those for the single criterion case. The existing methods are mainly genetic, tabu search, and simulated annealing based algorithms [7, 27, 31, 17].

In this paper we follow this line of research, aiming to get an accurate approximation of the whole set of non-dominated solutions of the bi-criteria MDKP problem. Our interest concerns large size instances. The Scatter Search (SS) meta-heuristic due to Glover [11] is used together with the surrogate relaxation technique.

SS is a population based evolutionary method, which exploits the knowledge of the problem to create new, and hence better solutions from the combination of existing ones. The fact that the relevant information regarding the optimal solution is embedded in a diversified subset of "elite" solutions is one of the fundamentals of SS. Taking multiple solutions into account as a foundation for creating new ones and using heuristics which combine them through mechanisms that promote diversity and quality, SS thus enhances the exploration of the information not contained in each solution individually. The usual process for solving a problem by means of creating progressively better solutions is divided into five components [11]: (1) the diversification generation method (which creates a collection of trial solutions), (2) the improvement method (which transforms the trial solutions into enhanced ones, and usually restores feasibility), (3) the reference set update method (which maintains the reference set with the best solutions according to certain criteria), (4) the subset generation method (which creates subsets of solutions from the reference set), and (5) the solution combination method (which combines solutions from each subset, thus creating new ones).

The technique of surrogate relaxation [10, 5] is used to convert the multi-constraint knapsack into the well-known single constraint knapsack by generating adequate surrogate multipliers. The aggregation of the constraints can give a good information about the efficient solutions of the original problem, working as a powerful tool to guide the search of those solutions. A modified version of the method due to Gomes da Silva et al. [13] is applied to the surrogate problem.

The rest of the paper is organized as follows: Section 2 presents the bi-criteria MDKP; 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. The Bi-Criteria Multi-Dimensional Knapsack Problem

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

$$\begin{aligned}
\max z_1(x_1,\ldots,x_j,\ldots,x_n) &= \sum_{j=1}^{n} c_j^1 x_j,\\
\max z_2(x_1,\ldots,x_j,\ldots,x_n) &= \sum_{j=1}^{n} c_j^2 x_j\\
\text{subject to:}\quad & \sum_{j=1}^{n} a_{ij}x_j \leqslant b_i, \quad i=1,\ldots,m,\\
& x_j \in \{0,1\}, \quad j=1,\ldots,n,
\end{aligned} \tag{1}$$

where, c_j^1 and c_j^2 represents the value of item j in the criteria 1 and 2, respectively, $x_j = 1$ if item j, $(j = 1,\ldots,n)$ is included in the knapsack and $x_j = 0$ otherwise, a_{ij} means the weight of item j in knapsack constraint i $(i = 1,\ldots,m)$ and b_i is the overall amount of the resource i.

We assume that c_j^1, c_j^2, b_i and a_{ij} are positive integers and that $a_{ij} \leqslant b_i$ with $\sum_{j=1}^{n} a_{ij} > b_i$.

Constraints $\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i$, $i = 1,\ldots,m$ and $x_j \in \{0,1\}$, $j = 1,\ldots,n$ define the feasible region in the *decision space*, and their image when applying 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 a feasible solution, y, such that $z_k(x) \leqslant z_k(y)$, $k = 1, 2$ and $z_k(x) < z_k(y)$ for at least one k. 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).

Glover [10] introduced a relaxation technique called the surrogate relaxation. In this technique non-negative multipliers are used to aggregate difficult constraints.

Applying the surrogate relaxation to problem (1) for creating a linear combination of the m knapsack constraints, we have the so called *surrogate constraint*:

$$\sum_{i=1}^{m} u_i \sum_{j=1}^{n} a_{ij} x_j \leqslant \sum_{i=1}^{m} u_i b_i \tag{2}$$

which is equivalent to,

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} u_i a_{ij} \right) x_j \leqslant \sum_{i=1}^{m} u_i b_i. \tag{3}$$

Considering $w_j = \sum_{i=1}^{m} u_i a_{ij}$ and $W = \sum_{i=1}^{m} u_i b_i$, the surrogate problem can be written as follows,

$$\begin{aligned} \max z_1(x_1, \ldots, x_j, \ldots, x_n) &= \sum_{j=1}^{n} c_j^1 x_j, \\ \max z_2(x_1, \ldots, x_j, \ldots, x_n) &= \sum_{j=1}^{n} c_j^2 x_j \\ \text{subject to:} \quad & \sum_{j=1}^{n} w_j x_j \leqslant W, \\ & x_j \in \{0, 1\}, \quad j = 1, \ldots, n \end{aligned} \tag{4}$$

which is a single constraint knapsack problem.

The surrogate multipliers are used to build a surrogate constraint, transforming the problem into a single constraint one. There are several surrogate constraints, but the most adequate is the one which produces the correct combination of the resources, i.e., the one which produces a feasible region, as close as possible, to the one generated by the initial resource constraints.

By relaxing the integrality constraints in (1) an easier problem is obtained. However, in the presence of many constraints their surrogate relaxation can be interesting:

$$\begin{aligned} \max z_1(x_1, \ldots, x_j, \ldots, x_n) &= \sum_{j=1}^{n} c_j^1 x_j, \\ \max z_2(x_1, \ldots, x_j, \ldots, x_n) &= \sum_{j=1}^{n} c_j^2 x_j \\ \text{subject to:} \quad & \sum_{j=1}^{n} w_j x_j \leqslant W, \\ & x_j \in [0, 1], \quad j = 1, \ldots, n. \end{aligned} \tag{5}$$

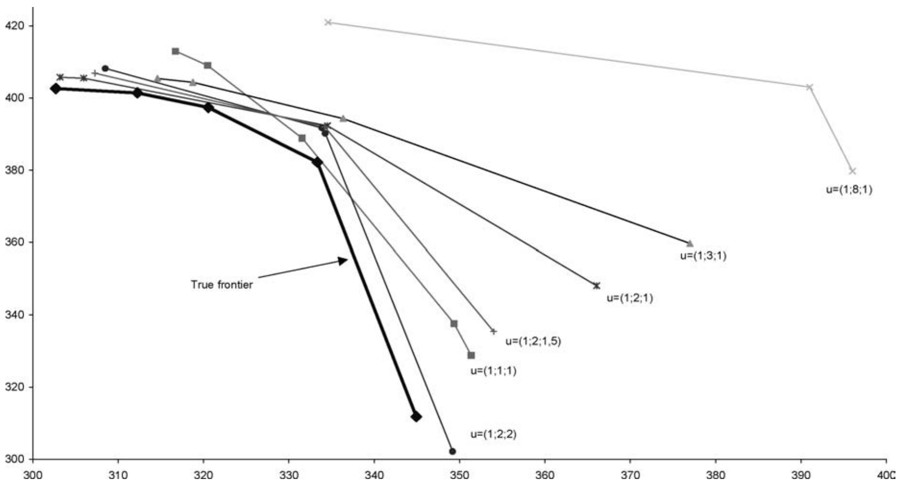

Figure 1. Surrogate frontiers for an instance with $n = 20$, $m = 3$.

Solving problem (5), i.e., determining the set of all the extreme efficient solutions, an upper frontier for problem (4) is obtained, which is obviously an upper frontier for the original problem (1). This frontier is placed above the one obtained by solving the linear relaxation of problem (1). We refer to the latter frontier as the *true upper frontier*.

However, with distinct surrogate multipliers different upper frontiers can thus be defined as it can be seen in Figure 1.

A set of multipliers which produces a frontier that is tight for one stretch of the true upper frontier could be a bad choice to another stretch of it. The objective is to find the set of multipliers which leads to a frontier which is as tighter as possible for all stretches of the true upper frontier.

One criterion that can be used to evaluate the tightness of a frontier is to consider the upper frontier which minimizes the sum of the maximum values of criteria z_1 and z_2. This criterion leads to the following particular problem:

$$\begin{aligned} h(u) = {} & \max\{z_1(x) : uAx \leqslant ub,\ x \in [0,1]^n\} + \\ & + \max\{z_2(x) : uAx \leqslant ub,\ x \in [0,1]^n\}, \end{aligned} \tag{6}$$

where A is the matrix with the original coefficients and u is the vector of the multipliers.

To distinguish the solutions in the two independent problems involved in $h(u)$ let us restate equation (6) as:

$$\begin{aligned} h(u) = {} & \max\{z_1(x) : uAx \leqslant ub,\ x \in [0,1]^n\} + \\ & + \max\{z_2(y) : uAy \leqslant ub,\ y \in [0,1]^n\} \end{aligned} \tag{7}$$

which is equivalent to,

$$h(u) = \max\{z_1(x) + z_2(y) : uAx \leqslant ub, uAy \leqslant ub,\ x, y \in [0,1]^n\}. \tag{8}$$

This criterion is particularly convenient because it is associated with a linear problem. When $h(u)$ decreases due to the simultaneous reduction of $z_1(x)$ and $z_2(x)$ then the frontier associated with the multipliers is indeed contained in a smaller area. In this case, the reduction of $h(u)$ corresponds to a decrease of the area where the frontier could be contained. Obviously, this is not always the case.

The associate surrogate multipliers are thus obtained by solving the dual surrogate problem:

$$\min_{u \in R_+^m} \{h(u)\}. \tag{9}$$

To determine the surrogate multipliers we consider the following iterative process, a subgradient-like method, which searches for a direction of potential decrease of the function $h(u)$.

```
**Procedure** Surrogate_Multipliers
begin
    $t \leftarrow 0$;
    Let $u^t \in R_+^m$ be a vector of multipliers such that $u^t \leftarrow \frac{1}{m}(1, 1, \ldots, 1)$;
    repeat
        Compute $h(u)$ and let $(x^{t*}, y^{t*})$ be an optimal solution;
        if $(Ax^{t*} - b \leqslant 0)$ and $(Ay^{t*} - b \leqslant 0)$ then stop;
        $g_{u^t} \leftarrow (Ax^{t*} - b) + (Ay^{t*} - b)$;
        Define $\theta \in R_+$ according to Rules 1 or 2 (below);
        $u^{t+1} \leftarrow u^t + \theta \frac{g_{u^t}}{\|g_{u^t}\|_2^2}$;
        if $(u_i^{t+1} < 0)$ then $u_i^{t+1} \leftarrow 0, \ i = 1, \ldots, m$;
        $u^{t+1} \leftarrow \frac{u^{t+1}}{\|u^{t+1}\|_1}$;
        $t \leftarrow t + 1$;
    until (stopping condition)
end
```

Notice that g_{u^t} is a direction of potential decrease of $h(u)$. In fact, as proved by Greenberg and Pierskalla [14] when searching for the optimal surrogate multipliers, one should neither decrease the multipliers of the infeasible constraints nor increase the multipliers of the feasible constraints. The direction g_{u^t} verifies these requirements in the cases (a) and (b) below, and can also verify it in the case (c):

(a) $(Ax^{t*} - b)_i \geqslant 0$ and $(Ay^{t*} - b)_i \geqslant 0$, where $(\gamma)_i$ is the component i of column vector γ.
In this case $(Ax^{t*} - b)_i + (Ay^{t*} - b)_i > 0$. Hence, $(g_{u^t})_i > 0$ and the multiplier of the infeasible constraint does not decrease.

(b) $(Ax^{t*} - b)_i \leqslant 0$ and $(Ay^{t*} - b)_i \leqslant 0$.

In this case $(Ax^{t*}-b)_i+(Ay^{t*}-b)_i < 0$. Hence, $(g_{u^t})_i < 0$ and the multiplier of the feasible constraint does not increase.

(c) $(Ax^{t*}-b)_i \times (Ay^{t*}-b)_i < 0$.
In this case $(Ax^{t*}-b)_i + (Ay^{t*}-b)_i$ could also be non-positive or nonnegative. If the dominant effect is the first (second), i.e., $(Ax^{t*}-b)_i+(Ay^{t*}-b)_i \leqslant 0$ $(\geqslant 0)$, then $(g_{u^t})_i \leqslant 0$ $(\geqslant 0)$, and it could be possible that the potential reduction of $h(u)$ due to the non-increase (non-decrease) of the multiplier of the feasible (infeasible) constraint is higher than the potential increase of $h(u)$ due to the non-increase (non-decrease) of the multiplier of the infeasible (feasible) constraint.

The parameter θ, which is the step-size, is initially equal to 2, and then it evolves according to the following rule $(t > 1)$:

RULE 1. $\theta = |u^{t-1}g_{u^t}|$.

This rule was suggested by Frévillle and Plateau [6] in the context of the subgradient algorithm for solving the surrogate and Lagrangian dual of the single criterion multi-dimensional knapsack problem.

However, when applied to the particular problem of minimizing $h(u)$ function, this rule leads frequently to a strong stagnation in the evolution of $h(u)$ (see Figure 2). We use an alternative rule which introduces variation in the step-size if the amount of infeasibility changes considerably. This new rule ensures a non-increase on the value of the step-size θ if the variation in the amount of the infeasibility increases. If the latter decreases then the step-size θ does not decrease. The rule is the following:

RULE 2. $\theta = \min\{2, \frac{1}{\pi^t}\}$, *where*

$$\pi^t = \begin{cases} \dfrac{|v_t - v_{t-1}|}{v_{t-1}} & \textit{if } v_{t-1} > 0, \\ 0.5 & \textit{if } v_{t-1} = 0, \end{cases}$$

v_t being defined as follows,

$$v_t = \sum_{\{i: a_i x^{t*} - b_i > 0\}} (a_i x^{t*} - b_i) + \sum_{\{i: a_i y^{t*} - b_i > 0\}} (a_i y^{t*} - b_i)$$

and a_i states for line i in matrix A.

Rule 2 performed well in practice (better than Rule 1).

In the above procedure, we used the maximum number of iterations as a stopping criterion.

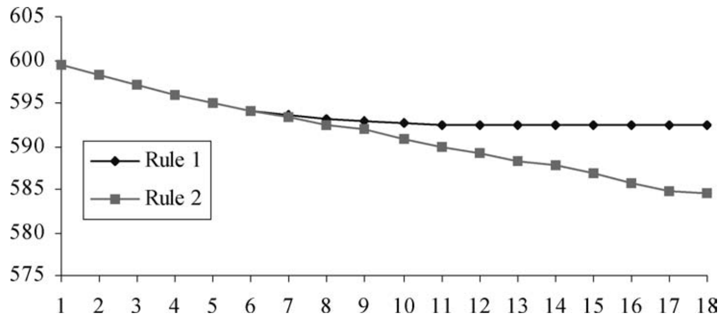

Figure 2. Evolution of $h(u)$.

EXAMPLE 1. Consider the following example of a bi-criteria multi-dimensional knapsack with 4 constraints and 10 variables:

$$\max z_1 = 1x_1 + 87x_2 + 28x_3 + 32x_4 + 38x_5 + 9x_6 + 8x_7 + 6x_8 + 92x_9 + 78x_{10}$$
$$\max z_2 = 4x_1 + 21x_2 + 68x_3 + 17x_4 + 43x_5 + 48x_6 + 85x_7 + 30x_8 + 37x_9 + 33x_{10}$$

subject to:

$$\begin{aligned}
&70x_1 + 85x_2 + 72x_3 + 31x_4 + 17x_5 + 33x_6 + 47x_7 + 25x_8 + 83x_9 + 28x_{10} \leqslant 246\\
&49x_1 + 15x_2 + 88x_3 + 29x_4 + 78x_5 + 98x_6 + 50x_7 + 89x_8 + 83x_9 + 3x_{10} \leqslant 291\\
&15x_1 + 15x_2 + 51x_3 + 3x_4 + 60x_5 + 1x_6 + 78x_7 + 66x_8 + 78x_9 + 71x_{10} \leqslant 219\\
&56x_1 + 21x_2 + 69x_3 + 60x_4 + 96x_5 + 65x_6 + 100x_7 + 25x_8 + 68x_9 + 30x_{10} \leqslant 295\\
&x_j \in \{0, 1\}, \quad j = 1, \ldots, 10
\end{aligned}$$

The application of the above procedure gives the results shown in Figure 2.

The multipliers which minimize $h(u)$ may not produce the "tightest" upper frontier. Hence, it is used a complementary criterion that is the area limited by the origin and the upper frontier generated by the three surrogate vectors $(\widehat{u}^1, \widehat{u}^2, \widehat{u}^3)$ associated with the three lowest values of $h(u)$. Considering,

$$\langle (z_1^1, z_2^1), (z_1^2, z_2^2), \ldots, (z_1^p, z_2^p) \rangle$$

as the sequence of extreme points in the upper frontier, the area can be expressed as:

$$R(\hat{u}^t) = \sum_{\alpha=1}^{p} \int_{l_{\alpha-1}}^{l_\alpha} f_t(z_1)\,\mathrm{d}z_1, \quad t = 1, 2, 3, \tag{10}$$

where, $l_\alpha = z_1^\alpha$, $\alpha = 1, \ldots, p$; $l_0 = 0$, and $f_t(z_1)$ is the function that expresses z_2 dependent of z_1 in the upper frontier associated with the surrogate multiplier vector u^t. An upper frontier is obtained by linking all the adjacent extreme points of problem (5). These points can be easily computed through a process of efficient pivoting over a simplex tableau with bounded variables (for details refer to Gomes da Silva et al. [12, 13]. With the extreme points it is easy to define the line and from this line the function $f_t(z_1)$ is derived.

This upper frontier will also be used to evaluate the quality of the solutions (Section 4) since it can provide a quantification of the error in assuming a potentially non-dominated solution as an exact non-dominated one.

3. A Scatter Search Method for the Bi-Criteria MCKP (BCSS)

With the use of surrogate multipliers the original problem is converted into a single constraint knapsack. For the latter we have recently experienced the problem of generating an approximation of the set of the non-dominated solutions of the bi-criteria case [12, 13]. Gomes da Silva et al. [13] proposed a scatter search based method which was applied to large size instances of the problem (a number of items up to 6,000 was considered). The method was supported by two fundamental properties of the solutions in the single criterion and bi-criteria instances, used as powerful tools for guiding the search of new solutions.

The first property comes from the single criterion problem and says that the optimal solution of the integer knapsack problem only differs from the continuous one in a very small number of items that are close to the fractional one.

The second property is based on the analysis of the exact efficient solutions of the bi-criteria instances, and states that an efficient solution can be obtained from another one by complementing the value (x_j is changed to $1-x_j$) of a small number of variables.

The results showed that the approximation to the upper frontier was quite good and an accurate description of the entire set of the non-dominated solutions could be obtained within small computational time. The method was structured according to the basic definition of the scatter search meta-heuristic: diversification method, improvement method, reference set update method, subset generation method, and solution combination method.

In this paper that method was adopted to deal with several constraints, taking into account the specificities of the new problem. In this sense, some of the BCSS components had to be considerably changed, namely the diversification method, the improvement method, and the combination method. The reference set update method and the subset generation method are the same. Aiming at preserving the global presentation of the method, all the components are described below.

3.1. DIVERSIFICATION METHOD

A non-stochastic procedure based on surrogate relaxation is used to generate a set of initial solutions. The initial set of solutions is obtained via the transformation of the multi-constraint problem into a single constraint one, using non-negative surrogate multipliers as described in Section 2. Regarding the continuous bi-criteria knapsack problem all the extreme solutions are found through efficient pivoting using the bi-criteria simplex method with bounded variables. Each of the extreme solutions has only one current basic variable, which is frequently a fractional variable.

The initial set of integer solutions contains thus the solutions obtained from the extreme solutions, including and excluding the item associated with the basic variable. These (feasible or infeasible) integer solutions are located around the boundary of the surrogate frontier, which is, by construction, near to the frontier defined by the exact non-dominated solutions of the original problem (1).

3.2. IMPROVEMENT METHOD

In the *improvement method* feasibility is restored and solutions are enhanced. Restoring feasibility in the MDKP is a very easy task, since removing an item from the knapsack reduces the infeasibility. However, selecting the most efficient process to restore feasibility is not so trivial. Several techniques can be found in the literature: profit-to-weight ratio based on surrogate relaxation [3]; profit-to-weight ratio considering the most violated constraint [15]; reduced prices based on Lagrangian relaxation [21]; maximum profit-to-weight ratio [30]; and, penalty functions [25, 20].

In our computational experiments the profit-to-weight ratio, considering the most violated constraint, performed better than any of the other strategies.

The infeasible solutions from the initial set are projected into the feasible region using that repair method considering the criteria z_1 and z_2 separately. The repair method searches for the most relatively violated constraint and remove the item with the lowest profit-to-weight ratio in that constraint, according to criteria z_1 and z_2. If the solution remains infeasible then the process is repeated.

The pseudo-code of the procedure is presented below.

Let x be an integer infeasible solution and k the selected criterion ($k = 1, 2$).

```
**Procedure** Remove_Items($x, k$)
**begin**
    $\Psi \leftarrow \{i : a_i x - b_i > 0; i = 1, \dots, m\}$;   {Violated constraints}
    **while** ($\Psi \neq \emptyset$) **do**
    **begin**
        {Search the most relatively violated constraint}
        $i^* \leftarrow \arg\max_{i \in \Psi} \left\{ \frac{a_i x - b_i}{b_i} \right\}$;
        {Search the item with the lowest ratio}
        $j^* \leftarrow \arg\min_{j=1,\dots,n} \left\{ \frac{c_j^k}{a_{i^*j}} : x_j = 1 \right\}$;
        $x_{j^*} \leftarrow 0$;   {Remove item}
        $\Psi \leftarrow \{i : a_i x - b_i > 0; i = 1, \dots, m\}$;   {Update $\Psi$ }
    **end**
**end**
```

After obtaining a feasible solution the insertion of additional items is investigated. The surrogate problem is used to identify the items that should be considered first. Items are inserted according to non-increasing order of the profit-to-weight ratios in the surrogate problem whenever any of the original constraints is not violated.

The last improvement of the solution is made by trying to replace the item last inserted by another one with a greater value on the considered criterion. The procedure can be presented as follows:

```
Procedure Add_Items($x, k$)
begin
    Consider $\frac{c_1^k}{w_1^k} \geqslant \cdots \geqslant \frac{c_j^k}{w_j^k} \geqslant \cdots \geqslant \frac{c_n^k}{w_n^k}$;   {Consider the items ordered}
    $last \leftarrow 0$;
    for ($j = 1$ to $n$) do
        if ($x_j = 0$) then
        begin
            $x_j \leftarrow 1$;
            if ($Ax > b$) then $x_j \leftarrow 1$; else $last \leftarrow j$;
        end
    if ($last > 0$) then
    begin
        $j^* \leftarrow \arg \max_{j=last+1,\dots,n} \left\{c_j^k : c_j^k > c_{last}^k, x_j = 0, A(x + e_j - e_{last}) \leqslant b\right\}$;
        if ($j^*$ exists) then $x_{j^*} \leftarrow 1, x_{last} \leftarrow 0$;
    end
end
```

In the above procedure e_j is a vector of size n with all the components equal to zero except component j which is equal to 1.

The improvement method transforms non-integer solutions obtained with the resolution of the continuous surrogate relaxation (5). The above procedures are used to define the improvement method for this kind of solutions.

Let x be a solution, β be the index of the basic variable (usually fractional), and *Initial_list* a list where all the obtained solutions are saved. The improvement method is as follows,

```
Procedure Improvement_Method($x, \beta$)
begin
    for ($k = 1$ to 2) do   {Number of criteria}
    begin
        $x^0 \leftarrow x$;
        $x_\beta^0 \leftarrow 0$;
        Add_items($x^0, k$);
        $x^0_\beta \leftarrow 1$;
        **Add_items**(**Remove_items**($x^0, k$), $k$);
        Add solutions to *Initial_list*;
    **end**
**end**
```

3.3. REFERENCE SET UPDATE METHOD

The *reference set update method* is usually composed of solutions that incorporate both quality and diversity features. In multiple criteria problems the attribute quality is not so obvious since there is not a unique function to evaluate the solutions. We use the concept of efficient solution as a way to incorporate high quality solutions. The diversity is incorporated with the use of the Hamming distance between each solution and a reference solution, x^{ref}, which is considered to be the one with the highest value on z_2 (it could also be z_1). This distance, d^{ref}, is computed for all the solutions in the set of potentially non-dominated solutions:

$$d_k^{\text{ref}} = \|x^{\text{ref}} - x^k\| = \sum_{j=1}^{n} |x_j^{\text{ref}} - x_j^k|, \quad k = 1, \ldots, |\tilde{X}|. \tag{11}$$

To define the reference set R with cardinality $|R|$ the solutions in $\tilde{X}$ are ordered according to non-decreasing values of d^{ref}. A number of $|R|$ groups with approximately the same number of solutions are constructed and from each group the mean solution is selected.

3.4. SUBSET GENERATION METHOD

The *subset generation method* defines the solutions to be combined in order to create new ones. The subsets are defined by all the pairs of consecutive solutions from the reference set. Hence, if the reference set has cardinality $|R|$ then $|R - 1|$ subsets are considered. Since 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, risking to lose some alternative solutions, i.e., different solutions in the decision space with the same image in the criteria space.

3.5. SOLUTION COMBINATION METHOD

The combination method explores paths between solutions of each subset. Let $\{x^0, x^1\}$ be a given subset. New solutions are obtained by incorporating features of x^1 in x^0 when the path $x^0 \rightarrow x^1$ is considered (x^0 is named the *initiating* *solution*, and x^1 the *guiding solution* in the nomenclature of Glover [11]), and by incorporating features of x^0 in x^1 when the reverse path is taken into account (x^0 is now the guiding solution, and x^1 the initiating one).

Consider for all $j = 1, \dots, n$,

$$(s_{x^0x^1})_j = \begin{cases} 1, & \text{if } x_j^0 = x_j^1 = 1, \\ 0, & \text{if } x_j^0 = x_j^1 = 0, \\ *, & \text{otherwise.} \end{cases}$$

The positions marked with "$*$" are called the *free positions* in the combination of x^0 and x^1, and the others are called the *fixed positions*.

The combination method investigates the possible change of all the free positions individually. This may imply changing some fixed positions, once feasibility of the change will always be required. Only two items are involved in the changes. This means that the combination method only searches for new potentially efficient solutions around a small neighborhood of x^0. This is a very important feature in order to keep a low computational time, but it is also justified by the fact that when many constraints are present in the problem and the residual capacity available is small it is usually difficult that more than one item can be included in all the knapsacks due to the variations in the coefficients of the matrix A.

When an item, corresponding to a free position, has the value 1 (0) in the *guiding solution* then the combination method tries to change the value of this item in the *initiating solution* by simply removing (inserting) just another item. The criteria z_1 and z_2 are used separately to search for the most profitable change. Hence, at the most, two new solutions are built.

Since the path from x^0 to x^1 is different of the path from x^1 to x^0, the solutions are swapped and once again the above combination procedure is applied. In this way, new regions of the decision space will be investigated, enabling the creation of new solutions.

Finally, note that the combination method only finds improved solutions: they are feasible and locally the best ones. That is the reason why in the BCSS method they are not subject to improvement.

The obtained solutions are put in a list of potentially non-dominated solutions, $\tilde{X}$, if they are not dominated by any of the solutions of that list. If the solution is inserted in $\tilde{X}$ then all the solutions that are dominated by it are removed. This process is named Update $\tilde{X}$.

The pseudo-code of the combination method is presented below.

```
**Procedure** Combination_Method($x^0$, $x^1$)
**begin**
    $F_1^0 \leftarrow \{j : x_j^0 \neq x_j^1, x_j^0 = 1, j = 1, \dots, n\}$;   {Items to be removed}
    $F_0^0 \leftarrow \{j : x_j^0 \neq x_j^1, x_j^0 = 0, j = 1, \dots, n\}$;   {Items to be inserted}
    $l \leftarrow 1$;
    while ($l \leqslant |F_1^0|$) do
    begin
        $j \leftarrow$ element $l$ of $F_1^0$;
        for ($k = 1$ to 2) do    {Number of criteria}
        begin
            $t^* \leftarrow \arg \max_{t \in \{1,\dots,n\} \setminus F_1^0} \{c_t^k : A(x^0 + e_t - e_j) \leqslant b\}$;
            if ($t^*$ exists) then $y \leftarrow x^0 + e_{t^*} - e_j$, Update $\tilde{X}$ with $y$;
        end
        $l \leftarrow l + 1$;
    end
    $l \leftarrow 1$;
    while ($l \leqslant |F_0^0|$) do
    begin
        $j \leftarrow$ element $l$ of $F_0^0$;
        for ($k = 1$ to 2) do    {Number of criteria}
        begin
            $t^* \leftarrow \arg \max_{t \in \{1,\dots,n\} \setminus F_0^0} \{c_t^k : A(x^0 + e_t - e_j) \leqslant b\}$;
            if ($t^*$ exists) then $y \leftarrow x^0 + e_{t^*} - e_j$, Update $\tilde{X}$ with $y$;
        end
        $l \leftarrow l + 1$;
    end
end
```

In the above algorithm, e_j is an n-dimensional vector with component j equal to 1 and all the others equal to 0.

3.6. THE OVERALL SCATTER SEARCH METHOD

The approach, stated in the procedure below, starts by applying the surrogate multipliers procedure. A vector of surrogate multipliers is thus obtained, and then used to transform the multi-constraint problem into a single constraint one. Considering the linear relaxation of the surrogate problem, the bi-criteria simplex method with bounded variables is applied to get the efficient extreme points, through a process of efficient pivoting [12]. In this process a weighted-sum function of the two criteria is used ($\lambda z_1 + (1-\lambda) z_2$, with $\lambda \in]0, 1[$). Each extreme efficient solution is associated with a range of weights. Adjacent ranges are associated with adjacent extreme efficient solutions. The Improvement Method is applied to each extreme efficient solution and the found (improved) solutions are added to the *Initial_list*, which are used to update the set of the potentially efficient solution, $\tilde{X}$. The main loop of the procedure below is composed of the Reference Set Update Method, the Subset

Generation Method, and the Combination Method. It is repeated till the stopping criterion is fulfilled.

From the Reference Set Update Method, it is obtained a reference set, R, from which several subsets are built. Then the Combination Method is applied to each subset if it was not already analyzed.

In order to avoid the repeated analysis of the same subset, the images (in the criteria space) of the solutions contained in it, are stored in the *Tabu_list*, that is searched before the Combination Method is applied.

```
Procedure Overall_Scatter_Search_MDKP
begin
    Initialize $\tilde{X} \leftarrow \emptyset$, Tabu_list $\leftarrow \emptyset$, Initial_list $\leftarrow \emptyset$;
    {Transform the multi-constraint problem into a single constraint one}
    Apply the procedure Surrogate_Multipliers;
    Use the determined multipliers to aggregate the constraints;
    Build the linear relaxation of the surrogate problem and maximize $z_2$;
    Let $x^0$ be the found optimal solution;
    Let $\beta^0$ the index of the basic variable in $x_0$;
    Improvement_Method($x^0$, $\beta^0$);
    Update $\tilde{X}$ with the Initial_list;
    $k \leftarrow 0$;
    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;
        Let $\beta^{k+1}$ denote the index of the basic variable in $x_{k+1}$;
        Improvement_Method($x^{k+1}$, $\beta^{k+1}$);
        $k \leftarrow k + 1$;
    end
    Terminate $\leftarrow$ false;
    while (Terminate = false) do
    begin
        Apply the Reference_Set_Update_Method;
        Let $R$ be the reference set obtained;
        Apply the Subset_Generation_Method;
        Let nsub be the number of created subsets;
        $k \leftarrow 1$;
        while ($k \leqslant$ nsub) do
        begin
            Let $x^0$ and $x^1$ be two solutions of subset $k$;
            if ((($z_1(x^0), z_2(x^0)$), ($z_1(x^1), z_2(x^1)$)) $\notin$ Tabu_list) then
            begin
                Combination_Method($x^0$, $x^1$);
                Insert $((z_1(x^0), z_2(x^0)), (z_1(x^1), z_2(x^1)))$ in *Tabu_list*;
            **end**
            $k \leftarrow k + 1$;
        **end**
        **if** (Stopping Condition verified) **then** *Terminate* $\leftarrow$ *true*;
    **end**
**end**
```

In the computational experiments (Section 4) it is used a maximum number of iterations as the stopping condition.

3.7. GRAPHICAL EXAMPLE

Let us illustrate graphically the behavior of the proposed BCSS method with an instance of the bi-criteria MDKP with 100 items and 10 constraints. The method starts by determining the surrogate frontier for the original problem (the line in Figure 3). All the extreme points of that frontier (in general non-integer points) are manipulated, giving rise to integer solutions. In the criteria space this solutions are placed below the frontier if the critical item is removed from the solution, or are placed above if the critical item is included in the solution (Figure 3). Using the procedure *Improvement Method* (Section 3.2) the integer points are projected into the feasible region of the original problem (Figure 4). This procedure pushes the initial points to the interior of the convex hull of the feasible space. Through successive iterations of BCSS new solutions are created from the initial ones. The new solutions reduce the gap between the upper frontier and the initial points (Figure 5). Figure 6 shows all the process.

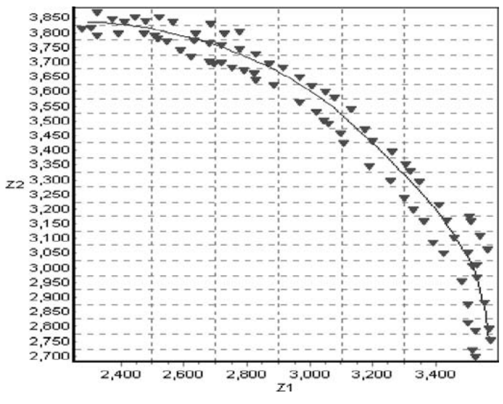

Figure 3. Surrogate frontier and initial solutions.

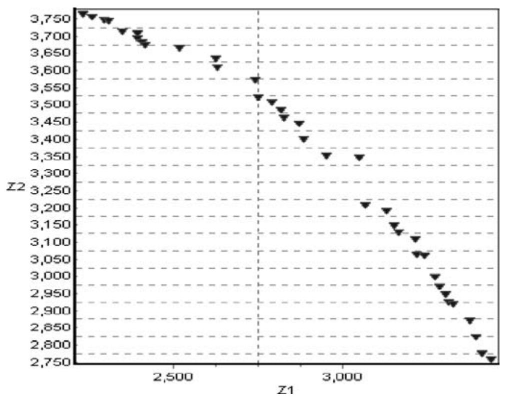

Figure 4. Improvement of the initial solutions.

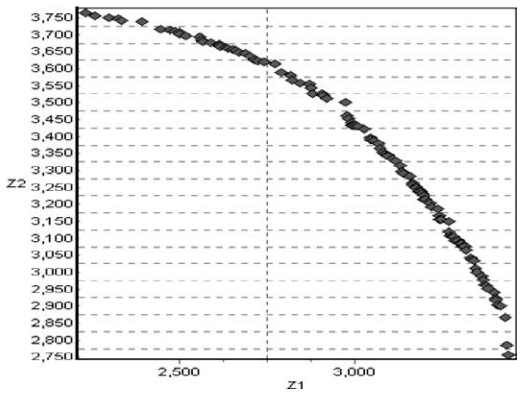

Figure 5. PNDS after several iterations.

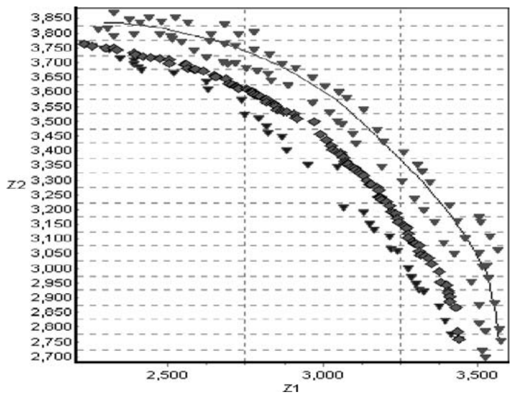

Figure 6. Integrated view.

4. Computational Experiments and Results

This section presents the computational experiments and the results obtained with the BCSS. The method is compared with other well-known multiple criteria meta-heuristics proposed by Zitzler and Thiele [31] and Jaszkiewicz [16], using their instances. Additional experiments are performed upon more complex instances. The quality of the results is evaluated taking into account two measures: proximity and diversity [30, 4, 2].

The computational experiments were performed on a Pentium 4 processor with 256 MB RAM and 40 GB hard disk. BCSS was implemented in Borland Delphi 4.

4.1. COMPARISON WITH OTHER META-HEURISTIC APPROACHES

Zitzler and Thiele [31] introduced a method named Strength Pareto Evolutionary Algorithm (SPEA). SPEA combines three techniques common to other meta-heuristics: (1) storage of all the non-dominated individuals (solutions) in an external set, (2) assignment of the scalar fitness according to the concept of the Pareto dominance, and (3) use of clustering to reduce the number of individuals in the external set. The fitness of a population member is determined only from the individuals in the external set. The latter is also used to select individuals for combination. The diversity of the population is enhanced by the use of a new Pareto based niching method.

Jaszkiewicz [16] proposed a multiple objective genetic local search algorithm (MOGLS). The algorithm is based on the fact that all the non-dominated solutions can be obtained by optimizing weighted Tchebycheff functions. Through the iterations of the algorithm several weighted Tchebycheff functions are built at random and the solutions (obtained by combination using a uniform crossover) are modified in order to improve these functions. The selection is based on the value of the solutions in the built random function. The solutions are selected from a set composed of the best ones according to the random function.

Both SPEA and MOGLS were applied to instances with a number of items up to 750 with 2, 3 and 4 objective functions and an equal number of constraints. Jaszkiewicz [16] showed that the developed MOGLS performed better than the procedures presented in Zitzler and Thiele [31], even with the SPEA.

Below are presented the results obtained with SPEA, MOGLS and BCSS using three bi-criteria instances (available at http://www.tik.ee.ethz.ch/~zitzler/testdata.html) employed by Zitzler and Thiele [31] and tested by Jaszkiewicz [16]. The results refer only to the first run of the above mentioned methods.

The BCSS runs 50 iterations of the procedure presented in Section 2 to determine the surrogate multipliers. The time spent is included is the total CPU time presented in Table I and the number of solutions in the reference set was fixed at 50.

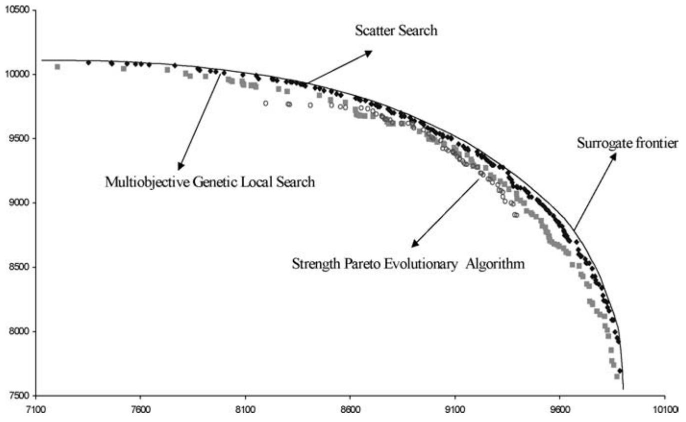

Figure 7. $n = 250$; $m = 2$.

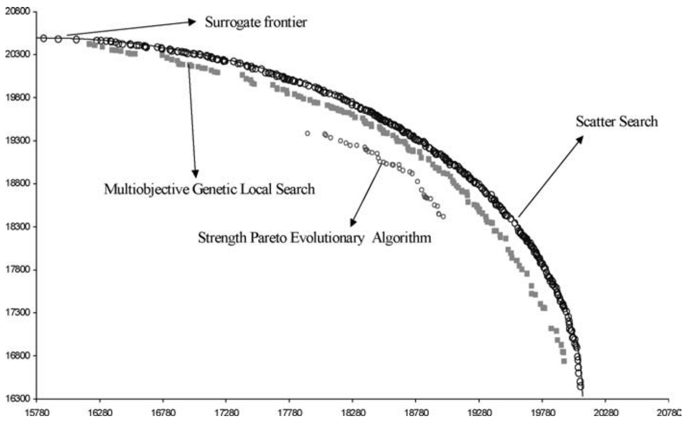

Figure 8. $n = 500$; $m = 2$.

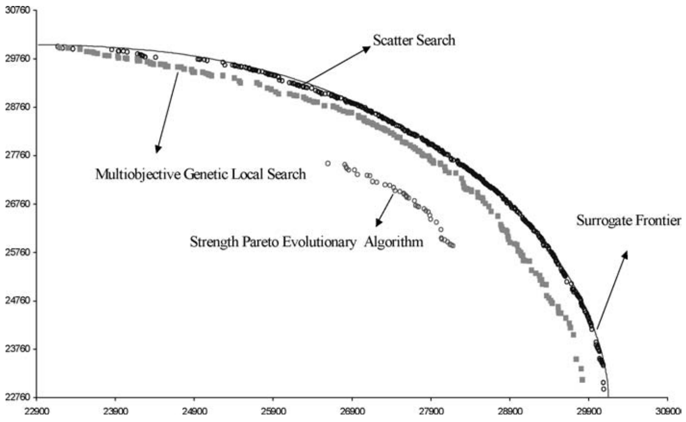

Figure 9. $n = 750$; $m = 2$.

Table I. PNDS and running times

	n $(m = 2)$	SPEA	MOGLS	BCSS	CPU (s) -BCSS
	250	60	105	153	1.1
PNDS	500	37	140	295	3.63
	750	41	182	404	8.02

All the solutions obtained with SPEA and MOGLS are dominated by the ones computed with BCSS. Empirically it can also be seen that the solutions of BCSS are well dispersed along the upper frontier.

The solutions obtained with BCSS are extremely close to the surrogate frontier, which represent the theoretical limit of their existence. The number of potentially non-dominated solutions is also larger giving an higher characterization of the set of the non-dominated solutions set (Table I).

4.2. RESULTS FOR LARGE SIZE INSTANCES

The above instances are very simple because the number of items is small and the number of constraints is only two. Large size instances are now considered. The limit of an hour of computation was fixed. This limit of time was enough to "solve" instances with a number of items between 1,000 and 3,000 with a number of the constraints from 10 to 100 (for $n = 1000, 2000$). The coefficients are integer numbers randomly and uniformly generated within the range [1, 100] and each knapsack constraint capacity is 50% of the sum of their weights. For each problem size, 15 instances were generated. For each instance, the BCSS ran 30 iterations. The optimal surrogate multipliers were determined in 50 iterations of the procedure presented in Section 2, and the number of solutions in the reference set was fixed at 30.

The quality of the approximation is evaluated taking into account two features: proximity and diversity. Considering the first feature, the L_∞ metric is used to determine the nearest point, (z_1^*, z_2^*), in the surrogate upper frontier of each PNDS, (z_1^+, z_2^+). 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^* is calculated using $f(z)$:

$$\frac{f(z^*) - f(z^+)}{f(z^*)} \times 100. \tag{12}$$

Diversity is evaluated based on two measures: (1) the standard deviation of the Euclidean distances between consecutive PNDS in the criteria space, and is similar to the proposed by Schott [24] the standard deviation of the number of solutions *per* region: the surrogate upper frontier is used to divide the criteria space into

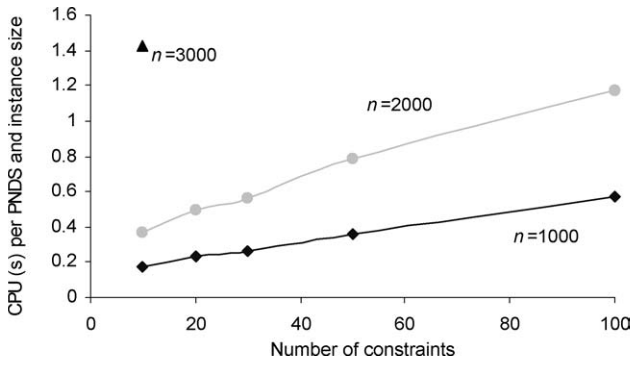

Figure 10. CPU (s) *per* PNDS.

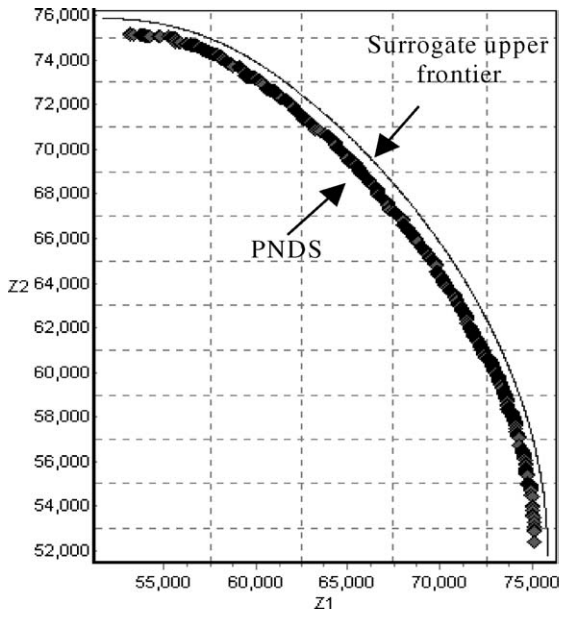

Figure 11. Applicationof BCSS to an instance with $n = 2000$ and $m = 20$.

regions. To do this, the range of criterion z_1 was divided in 20 equal size intervals, that projected into the space z_1z_2 give rise to 20 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 the greater the dispersion of the obtained solutions along the upper frontier.

Table II shows the results concerning the 165 instances. Column 1 refers to the number of items and constraints; column 2 concerns the computational time in seconds; column 3 shows the number of potentially non-dominated solutions; columns 4, 5, 6 and 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. In order to better

Table II. Results for large size instances

(1*a*)	(1*b*)		(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
n	*m*		CPU	PNDS	Average	Max.	Min.	STD	STD-C	STD-D
1000	10	Average	167.06	931.93	0.8335	1.3229	0.5206	0.0015	3.63	40.06
		Max	202.23	1011.00	1.2580	1.8454	0.7927	0.0030	4.29	51.48
		Min	150.28	894.00	0.4484	0.8559	0.2066	0.0006	2.97	25.76
		STD	14.82	33.39	0.2222	0.2895	0.1563	0.0005	0.32	5.91
	20	Average	242.44	1033.07	1.2710	1.8202	0.9445	0.0019	3.86	38.12
		Max	260.79	1111.00	1.7257	2.2639	1.3698	0.0034	4.64	51.24
		Min	214.93	932.00	0.7616	1.4641	0.5042	0.0010	3.37	29.02
		STD	13.32	49.04	0.2849	0.2075	0.2835	0.0006	0.32	5.59
	30	Average	304.58	1163.73	1.6818	2.2547	1.3067	0.0021	4.06	33.00
		Max	339.50	1289.00	1.9985	2.8797	1.6717	0.0047	4.64	41.52
		Min	280.45	1098.00	1.3889	1.9099	0.9285	0.0010	3.36	21.64
		STD	15.03	55.91	0.1963	0.2972	0.2242	0.0011	0.36	5.78
	50	Average	468.19	1300.87	2.0030	2.4987	1.6063	0.0019	4.23	28.66
		Max	545.69	1420.00	2.4149	2.9557	1.9769	0.0023	4.78	39.85
		Min	433.09	1165.00	1.8100	2.1639	1.3393	0.0013	3.94	20.19
		STD	33.67	74.71	0.1443	0.2126	0.1522	0.0004	0.26	5.84
	100	Average	819.74	1422.67	2.5125	2.9744	2.1349	0.0016	4.36	28.93
		Max	916.46	1605.00	2.8910	3.3407	2.4587	0.0029	5.16	41.57
		Min	734.95	1282.00	2.1762	2.5500	1.9341	0.0008	3.56	23.15
		STD	51.36	86.67	0.2004	0.2508	0.1458	0.0006	0.46	5.66
2000	10	Average	567.02	1536.00	0.6488	1.0253	0.3529	0.0022	4.68	55.07
		Max	639.06	1667.00	1.0293	1.4752	0.6591	0.0130	5.18	73.31
		Min	517.40	1234.00	0.4115	0.6745	0.1922	0.0007	4.07	41.45
		STD	32.51	109.62	0.1673	0.1771	0.1218	0.0029	0.25	8.01
	20	Average	870.51	1762.20	1.0719	1.4989	0.7213	0.0017	4.77	48.44
		Max	946.81	1885.00	1.3278	1.7572	0.9566	0.0032	5.47	60.51
		Min	735.40	1630.00	0.8412	1.0483	0.4835	0.0009	3.49	38.01
		STD	53.00	64.09	0.1556	0.1885	0.1364	0.0006	0.44	5.43
	30	Average	1108.44	1970.87	1.2209	1.6759	0.8928	0.0017	5.14	44.34
		Max	1369.13	2146.00	1.7023	2.5905	1.0973	0.0041	5.73	55.16
		Min	967.29	1753.00	1.0386	1.4058	0.6505	0.0008	4.45	36.07
		STD	100.16	101.79	0.1774	0.2915	0.1114	0.0008	0.37	5.06

Table II. (Continued)

(1*a*) *n*	(1*b*) *m*		(2) CPU	(3) PNDS	(4) Average	(5) Max.	(6) Min.	(7) STD	(8) STD-C	(9) STD-D
	50	Average	1796.97	2283.00	1.5407	1.9478	1.2680	0.0014	5.50	38.27
		Max	2173.62	2487.00	1.7340	2.5385	1.4740	0.0038	6.24	45.80
		Min	1542.31	2147.00	1.2199	1.6880	1.0170	0.0008	4.47	32.29
		STD	166.52	90.05	0.1390	0.1944	0.1381	0.0007	0.41	3.52
	100	Average	3110.64	2649.79	1.7660	2.1329	1.4918	0.0013	6.18	34.96
		Max	3395.82	2883.00	1.9974	2.5412	1.6809	0.0020	7.09	42.83
		Min	2749.45	2441.00	1.5075	1.7533	1.3270	0.0008	5.33	25.25
		STD	205.27	130.01	0.1424	0.2246	0.1113	0.0004	0.46	5.07
3000	10	Average	3113.83	2182.73	0.5421	0.8846	0.2841	0.0012	5.46	64.82
		Max	3580.70	2404.00	0.7524	1.2123	0.5140	0.0021	6.34	80.75
		Min	1944.80	1896.00	0.2568	0.6258	0.1011	0.0009	4.43	51.13
		STD	450.13	160.99	0.1341	0.1506	0.1090	0.0004	0.47	10.10

understand the results, see for instance the meaning of the value 1.3229 (first line, fifth column). It means the average (concerning 15 instances) of the maximal L_∞ distances.

Results show that the average percentage gap is less than 2.6% (column 4), which is quite satisfactory attending to the fact that the comparisons are made with the surrogate upper frontier which is above the one derived from the linear relaxation. The percentage gap tends to raise, for a given number of items, with the increase of the number of the constraints. This is because the surrogate upper frontier is less tight. Given a number of constraints an inverse relationship is observed between the percentage gap and the number of items. Concerning diversity, the solutions are well dispersed along the criteria space (column 8) and the standard deviation of the Euclidean distance between consecutive solutions is also small (column 9). The average CPU time is an increasing function with the number of constraints and items (column 2). A more interesting observation concerns the computational time required for determining each PNDS. Figure 10 shows that it is almost a linear function of the number of constraints, and the rate of its increase is higher in instances with a greater number of items. This is essentially due to the greater number of comparisons required by the BCSS, particularly in the procedures for recovering the feasibility and in the improvement of the solutions.

The number of PNDS generated by the BCSS increases with the number of constraints, for a given number of items (Table II). This is mainly justified by the design of the combination method, which explores the dissimilarity of the solutions in the decision space. So, as an higher number of constraints induces more dissimilar solutions, the combination method generates more solutions, increasing the probability of obtaining an higher number of PNDS.

Figure 11 presents an example of the result of the application of BCSS method to an instance with 20 constraints and 2,000 items. This figure illustrates the proximity of the surrogate upper frontier and the set of the potentially non-dominated solutions.

5. Conclusions and Future Research

In this paper we proposed a scatter search based method to deal with the bi-criteria MDKP. In the diversification component the surrogate relaxation was used to convert the multi-constraint problem into a single constraint one. This is a new aspect in the context of other meta-heuristics applied to the multiple criteria MDKP. The critical part was the determination of the surrogate multipliers which reflect the correct combination of all the constraint resources of the problem. This is associated with a non-trivial problem. A subgradient-like procedure was developed to solve it. The BCSS method was organized according to the basic structure of the scatter search method. Specific procedures were built for each component, but a great influence can be identified from our previous work with the single constraint knapsack [13].

Comparisons with the MOGLS and SPEA meta-heuristics were made using the same instances. The BCSS showed to be superior since all the solutions dominate the ones obtained with the MOGLS and SPEA. Motivated by the results of the comparison, the BCSS was also tested in more complex instances. The results proved to be of high quality concerning proximity and diversity. We believe that the quality of the initial solutions was determinant in this result.

Nevertheless, several aspects can be improved: (1) the procedure for determining the surrogate multipliers was based on the minimization of the sum of the maximum values of each criterion. An alternative objective function could be developed. Additional experiments with a flexible stopping criterion may also be interesting; (2) the combination method uses a very strong structure for searching for new solutions. This is particularly adequate to keep the computational time low, but several interesting solutions were, for this reason, not determined. In this sense, a broader scope of the search in the combination of solutions can also be considered.

Acknowledgements

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

1. Chardaire, P., McKeown, G. and Maki, J.: Application of GRASP to the multi-constraint knapsack problem, in Boers et al. (eds), *Evoworkshop*, Lecture Notes in Comput. Sci. 2037, Springer-Verlag, Berlin, 2001, pp. 30–39.
2. Coello, C., Veldhuizen, D. and Lamont, G.: *Evolutionary Algorithms for Solving Multi-objective Problems*, Kluwer Acad. Publ., Dordrecht, 2002.
3. Chu, P. and Beasley, J.: A genetic algorithm for the multi-dimensional knapsack problem, *J. Heuristics* **4** (1998), 63–86.
4. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*, Wiley, New York, 2001.
5. Dyer, M.: Calculating surrogate constraints, *Math. Programming* **19** (1980), 255–278.
6. Frévillle, A. and Plateau, G.: An efficient preprocessing procedure for the multi-dimensional 0-1 knapsack problem, *Discrete Appl. Math.* **49** (1994), 189–212.
7. Gandibleux, X., Mezdaoui, N. and Fréville, A.: A tabu search procedure to solve multiobjective combinatorial optimization problems, in R. Caballero, F. Ruiz and R. Steuer (eds), *Proceedings of MOPGP96*, Springer-Verlag, Berlin, 1996, pp. 291–300.
8. Gavish, B. and Pirkul, H.: Efficient algorithms for solving multi-constraint zero-one knapsack problems to optimality, *Math. Programming* **31** (1985), 78–105.
9. Gilmore, P. and Gomory, R.: The theory and computation of knapsack functions, *Oper. Res.* **14** (1966), 1045–1074.
10. Glover, F.: Surrogate constraint duality in mathematical programming, *Oper. Res.* **23**(3) (1975), 434–451.
11. Glover, F.: Scatter search and path relinking, in D. Corne, M. Dorigo and F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, 1999, pp. 297–316.
12. Gomes da Silva, C., Figueira, J. and Clímaco, J.: An interactive procedure dedicated to the bi-criteria knapsack problem, Research Report 4, INESC-Coimbra, Portugal, 2003 (in Portuguese), http://www.inescc.pt/download/RR2003_04.pdf.
13. Gomes da Silva, C., Clímaco, J. and Figueira, J.: A scatter search method for bi-criteria knapsack problems, *European J. Oper. Res.* (2004), to appear.
14. Greenberg, H. and Pierskalla, W.: Surrogate mathematical programming, *Oper. Res.* **21** (1970), 924–939.
15. Hanafi, S. and Fréville, A.: An efficient tabu search approach for the 0-1 multi-dimensional knapsack problem, *European J. Oper. Res.* **106** (1998), 659–675.
16. Jaszkiewicz, A.: On the performance of multiple objective genetic local search on the 0/1 knapsack problem. A comparative experiment, Research Report RA-002/2000, Institute of Computing Science, Poznan University of Technology, Poznan, Poland, 2000.
17. Jaszkiewicz, A.: Genetic local search for multi-objective combinatorial optimization, *European J. Oper. Res.* **137** (2002), 50–71.
18. Jones, D., Mirrazavi, S. and Tamiz, M.: Multi-objective meta-heuristics: An overview of the current state-of-the-art, *European J. Oper. Res.* **137** (2002), 1–9.
19. Levenhagen, J., Bortfeldt, A. and Gehring, H.: Path tracing in genetic algorithms applied to the multiconstrained knapsack problem, in Boers et al. (eds), *Evoworkshop*, Lecture Notes in Computer Sci. 2037, Springer-Verlag, Berlin, 2001, pp. 40–49.
20. Loulou, R. and Michaelides, E.: New greedy-like heurisics for the multi-dimensional 0-1 knapsack problem, *Oper. Res.* **27**(6) (1979), 1101–1114.
21. Magazine, M. and Oguz, O.: A heuristic algorithm for the multi-dimensional zero-one knapsack problem, *European J. Oper. Res.* **16** (1984), 319–326.
22. Pirkul, H.: A heuristic solution procedure for the multi-constraint zero-one knapsack problem, *Naval Res. Logist.* **34** (1987), 161–172.
23. Sarin, S., Karwan, M. and Rardin, R.: Surrogate duality in a branch-and-bound procedure for integer programming, *European J. Oper. Res.* **33** (1988), 326–333.

24. Schott, J.: Fault tolerance design using single and multi-criteria genetic algorithms, Master's thesis, Massachusetts Institute of Technology.
25. Senju, S. and Toyoda, Y.: An approach to linear programming with 0-1 variable, *Management Sci.* **15**(4) (1968), 196–207.
26. Shih, W.: A branch and bound method for the multi-constraint zero-one knapsack problem, *J. Oper. Res. Soc.* **30** (1979), 369–378.
27. Ulungu, E., Teghem, J., Fortemps, Ph. and Tuyttens, D.: A MOSA method: A tool for solving multiobjective combinatorial optimization problems, *J. Multi-Criteria Decision Anal.* **8** (1999), 221–236.
28. Vasquez, M. and Hao, J.: A hybrid approach for the 0-1 multi-dimensional knapsack problem, in *Proceedings of IJCAI-01*, Seatle, Washington, 2001.
29. Weingartner, H. and Ness, D.: Method for the solution of the multi-dimensional 0-1 knapsack problem, *Oper. Res.* **15** (1967), 83–103.
30. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: Methods and applications, Ph.D. thesis, Swiss Federal Institute of Technology of Zurich, Zurich, Switzerland, 1999.
31. Zitzler, E. and Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto Approach, *IEEE Trans. Evolutionary Comput.* **3**(4) (1999), 257–271.