Solving the bi-objective multidimensional knapsack problem exploiting the concept of core

George Mavrotas[a], José Rui Figueira[b], Kostas Florios[c]

[a]National Technical University of Athens, Zographou Campus, 15780, Athens, Greece,
e-mail: mavrotas@chemeng.ntua.gr

[b]CEG-IST, Center for Management Studies, Instituto Superior Técnico, Technical University of Lisbon, Tagus Park, Av. Cavaco Silva, 2780 - 990 Porto Salvo, Portugal, e-mail: figueira@ist.utl.pt

[c]National Technical University of Athens, Zographou Campus, 15780, Athens, Greece,
e-mail: cflorios@central.ntua.gr

Abstract: This paper deals with the bi-objective multi-dimensional knapsack problem. We propose the adaptation of the core concept that is effectively used in single objective multi- dimensional knapsack problems. The main idea of the core concept is based on the "divide and conquer" principle. Namely, instead of solving one problem with *n* variables we solve several sub-problems with a fraction of *n* variables (core variables). The quality of the obtained solution can be adjusted according to the size of the core and there is always a trade off between the solution time and the quality of solution. In the specific study we define the core problem for the multi-objective multidimensional knapsack problem. After defining the core we solve the bi-objective integer programming that comprises only the core variables using the Multicriteria Branch and Bound algorithm that can generate the complete Pareto set in small and medium size multi-objective integer programming problems. A small example is used to illustrate the method while computational and economy issues are also discussed. Computational experiments are also presented using available or appropriately modified benchmarks in order to examine the quality of Pareto set approximation with respect to the solution time. Extensions to the general multi-objective case as well as to the computation of the exact solution are also mentioned.

1. Introduction

Over the last years, much work has been done in the single objective Multi-Dimensional Knapsack Problem (MDKP, see Freville, 2004 for a recent overview). In their recent textbook Kellerer et al., (2004, chapter 9) devote a whole chapter to multidimensional knapsack problem. A recent breakthrough in multidimensional knapsack problems is the concept of core as described by Puchinger et al. (2006). The concept of core for the conventional (single constraint) knapsack was already known from the mid 90s (see e.g. Pisinger,1995), but Puchinger et al. (2006) expanded it to the multidimensional case. More recently, the concept of core was addressed by Gomes da Silva et al. (2008) in bi-criteria, single dimensional knapsack problems.

The aim of our work is to synthesize the ideas of Puchinger et al. (2006) and Gomes da Silva et al. (2008) in order to define and apply the core issue in Multi-Objective Multi-Dimensional Knapsack Problem

(MOMDKP). Specifically, in this paper we deal with the simplest case the Bi-Objective Multi-Dimensional Knapsack Problem (BOMDKP) and we present some first results.

The idea is to decompose the BOMDKP to a series of single objective MDKP sub-problems following the ideas of Gomes da Silva et al. (2008). Then, exploiting the ideas of Puchinger et al. (2006), we define a core for each one of the MDKP sub-problems. Using only the core variables, we appropriately modify the BOMDKP problem and solve it, generating the complete Pareto set of each subproblem, using the Multi-Criteria Branch & Bound (MCBB) method (Mavrotas and Diakoulaki, 1998, 2005). MCBB is a general purpose method for solving Mixed Integer Multi Objective Linear Programming (MOMILP) problems of small and medium size. In the specific application MCBB is adjusted to problems with only integer (more specifically, binary) variables. It must be clarified that the obtained Pareto set corresponds to the specific core variables and not to the original BOMDKP. We repeat this procedure for all the MDKP subproblems and at the end we merge the obtained Pareto sets to obtain a representation of the Pareto set of the original problem. The size of the core problems is manageable with the MCBB method.

The basic idea behind the proposed method is the well known "divide and conquer" principle that is often used in OR techniques. The "divide and conquer" principle is especially beneficial in the case of the knapsack problem which is combinatorial explosive as an NP-hard problem. Instead of solving a problem with n variables we solve several problems with a fraction of n variables (e.g. it is much faster to solve 50 BOMDKP problems with 20 variables than 1 BOMDKP with 40 variables). As it will be shown in the results the computational economy for medium and big problems is significant, while the quality of the Pareto set representation is controllable: increasing the core size the quality of representation also increases. However there is always a trade-off between the computational time and the quality of representation.

Although the proposed method is originally designed as an approximate method (producing a representation of the Pareto set) it can be used as a good first step for the exhaustive production of the complete Pareto set as it is discussed in the last section of the paper. The exact solution of MOMDKP problems (complete Pareto set), is very useful as they can be used as reliable benchmarks for the widely used multiobjective metaheuristics. Moreover, the method with just small modifications can be extended to solve multi-objective knapsack problems with more than two objectives.

The structure of the paper is as follows. Following the introduction we define in Section 2 the core problem for the MOMDKP. In section 3 we present the algorithm for the BOMDKP and in section 4 we describe the application of the algorithm in detail using an educational example. In section 5 we introduce the concept of the adjusted core while in section 6 we present some computational results. Finally in section 7 we present the basic concluding remarks and some discussion about future research.

2. The core concept for the Multi-Objective Multi-Dimensional Knapsack Problem

In conventional knapsack problems with one objective function and one constraint, the core is a subset of items-variables with efficiencies (ratio of price to weight) that are similar to the efficiency of the break item. The core concept was the basic idea in the development of the most efficient algorithms for the knapsack problem. Recently, there has been defined the concept of core for (i) multidimensional knapsack problems (see Puchinger et al., 2006) and (ii) for bi-objective single dimension knapsack problems (see

Gomes da Silva et al., 2008). Our proposal is the definition and effective implementation of the core concept for the bi-ojective multidimensional knapsack problem. This extension is not trivial. We synthesize the two previous works and define the appropriate core concept on the bi-objective multidimensional knapsack problem.

The main idea is to exploit the "divide and conquer" principle which is so effective in operational research and more specifically in combinatorial problems (see e.g the decomposition algorithms or branch and bound). Instead of solving one big problem we solve a series of smaller problems synthesizing the results at the end, obtaining so the desired solution. As we will show, the intermediate results can be used to accelerate the whole process.

The main idea is briefly the following: According to Gomes da Silva et al. (2008) we initially solve the relaxed problem producing the set of Efficient Extreme Solutions (EES). Then we assign to every EES the corresponding weight interval according to Multi-Objective Linear Programming (MOLP) theory. For each one of the EES and using the proper weight coefficients to merge the two objective functions to one we transform the problem to single objective. Then we take advantage of the Puchinger et al. 2006, findings regarding the efficiencies and the core concept of the multidimensional knapsack problem. Namely, for each one of the EES we assign the appropriate core (according to Puchinger et al. (2006) dual efficiencies). We adjust the initial BOMDKP to the specific core variables and solve it with MCBB in order to generate the local (specific to this particular EES) Pareto set. We repeat this process for all the EES and at the end we merge the local Pareto sets to obtain a representation of the complete Pareto set of the initial problem.

The whole idea can be better expressed with an illustrative graph. In Figure 1 the Pareto front of the relaxed problem is represented by the line A,B,C,D. For each one of the four EES we compute the core (see next section). The variables that are outside the core are fixed either to 0 or 1 ("frozen" variables). The fixed part corresponds to points A', B', C' D'. The four BOMDKP that are solved with MCBB and involve only the core variables produce the relevant (locally) Pareto optimal solutions. For example from the core variables of the A-core, the locally Pareto optimal solutions A1, A2, A3 are produced. After all the EES are visited, the locally Pareto optimal solutions are merged to derive the representation of the Pareto set of the original BOMDKP. The whole process and the specific computational details are described thoroughly in the next sections with an educational example.

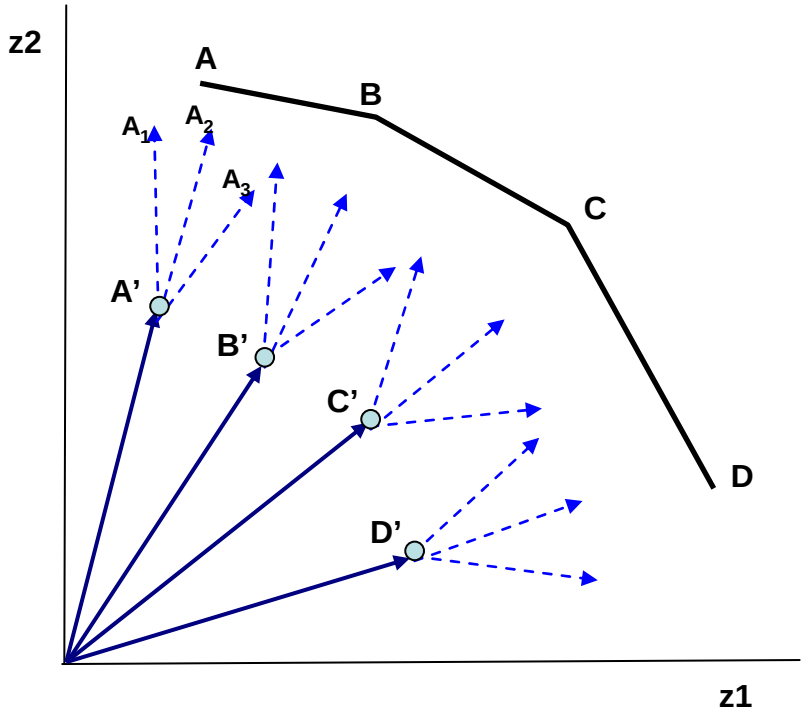

Figure 1: Graphical representation of the core concept in BOMDKP

3. The proposed algorithm

Assume the following BOMDKP problem:

$$\begin{aligned} &\max \ \textstyle\sum_{j=1}^{n} p_{kj} x_j \qquad k=1,2 \\ &st \\ &\textstyle\sum_{j=1}^{n} w_{ij} x_j \le c_i \qquad i=1...m \\ &x_j \in \{0,1\} \qquad j=1,...,n \end{aligned} \tag{1}$$

The proposed algorithm for applying the core concept in bi-objective multidimensional knapsack problems has the following steps:

- Step 1: Relax the binary variables ($x_j \in [0,1]$) and solve the resulted Multiple Objective Linear Programming (MOLP) problem which is the relaxation of the BOMDKP problem and is defined as RBOMDKP. Derive the Efficient Extreme Solutions (EES) of RBOMDKP using an appropriate method. Methods like ADBASE (Steuer, 1995) or EFFTREE (Mavrotas, 2000) can be used for the exhaustive enumeration of the efficient extreme solutions or a weighted sum algorithm for an approximation. The number of EES is R.
- Step 2: As it is well known from MOLP theory, every EES can be obtained as an optimal solution of an LP problem, where the objective function is a weighted sum of the MOLP objective functions. Therefore, at every EES corresponds a weight interval that can be easily calculated (see e.g. Steuer, 1989 page 124-125). Calculate the weight intervals that correspond to every EES (in the bi-objective case k=2 and λ_2=1-λ_1)

- Step 3: For every EES, which means for $r=1...R$, repeat Steps 3a-3h
  - Step 3a: Use the appropriate weights as computed in Step 2, create the LP with the weighted sum objective function and solve it.
  - Step 3b: Use the shadow prices (d_i, i=1…m) of the constraints in order to compute the dual efficiencies $e_j(duals)$ of the decision variables, following the Puchinger et al. (2006) terminology for single objective multidimensional knapsack (SOMDKP) problems. Specifically we have:

$$e_j(duals) = \frac{p_j}{\sum_{i=1}^{m} d_i w_{ij}} \tag{2}$$

  - Step 3c: Sort the variables in decreasing order according to their efficiencies e_j and find the split interval (the variables with e_j=1) and the center of the split interval (c) as described in Puchinger et al. (2006).
  - Step 3d: A core C_r is defined as the set of variables that are around c. It usually consists of $\delta \times n$ items to the left and right of the center of the split interval. The parameter δ determines the size of the core and is usually defined as a fraction of total number of variables, n (e.g. δ=0.1 means that $0.1 \times n$ variables to the left and to the right of the split interval center are included in the core). The size of the core is $2 \times \delta \times n+1$ variables. In this step, by defining δ we directly determine the size of the core.
  - Step 3e: Check if the core C_r is included in previous cores C_k (k=1…r-1). If yes, go back to the beginning of step 3 and proceed to next r (i.e. it is redundant to calculate the corresponding Pareto optimal solutions as they are already found). Otherwise, continue
  - Step 3f: Create the model BOMDKP(r) which contains only the core variables. This means that the variables that are left to the core are fixed to "1" and the variables that are on the right of the core are fixed to "0". Adjust the objective functions by incorporating the appropriate fixed terms (*lcore* is the set of variables on the left of the core) :

$$z_{kF} = \sum_{j \notin lcore} p_{kj} \tag{3}$$

Adjust also the right hand side (c_i) of the constraints as follows:

$$c_i' = c_i - \sum_{j \notin lcore} w_{ij} \tag{4}$$

The model BOMDKP(r) is called the *core subproblem* and has the following form:

$$\begin{aligned} &\max\ z_{kF} + \sum_{j \in core} p_{kj} x_j \qquad k=1,2 \\ &st \\ &\sum_{j \in core} w_{ij} x_j \le c_i' \qquad i = 1...m \\ &x_j \in \{0,1\} \qquad j \in core \end{aligned} \tag{5}$$

- Step 3g: Solve the core subproblem using the MCBB algorithm (Mavrotas and Diakoulaki, 1998; 2005) and generate all the corresponding Pareto optimal solutions (POS). These POS are defined as *locally* POS (LPOS) as they refer to the specific core subproblem (BOMDKP(r)) and not to the original BOMDKP problem (1).
- Step 3h: Store the LPOS solutions to the Incumbent List (IL) which is the list with the LPOS found so far (from previous core subproblems for r>1). Update the IL by deleting the LPOS that are found to be dominated.

- Step 4: End of the algorithm. The IL contains the globally Pareto optimal solutions for problem BOMDKP.

The flowchart of the algorithm is shown in Figure 2 while in Figure 3 the flowchart for the core definition is presented. As it was mentioned, the advantage of the proposed method relies on the improvement of solving *R* times problem (5) instead of solving once the problem (1).

In the remainder of the section some computational issues regarding the implementation of the algorithm are discussed. In step 1, the number of EES depends heavily on the number of variables *n* and constraints *m*. If the size of the problem is relatively large (e.g. several hundreds of variables and 10-30 constraints), the number of the EES may be in the order of thousands, which makes almost prohibitive the implementation of the next steps of the algorithm. In these cases an adequate and controllable approximation of the set of EES is recommended. This can be easily derived using multiple LP runs with diverse weighted sums of the objective functions. The inclusion check of every new core (step 3e) prevents us from redundant solutions of essentially the same problems. This computational economy due to this check is proved significant.

The algorithm is using MCBB for the generation of the complete set of the LPOS in the core subproblems in step 3f. In every iteration *r* with *r*>1, the already found LPOS vectors that exist in the incumbent list are used in the fathoming conditions of the MCBB algorithm. These vectors are used to check the fathoming condition in every intermediate node by comparing them with the ideal (non-feasible) vector of the node (in the same sense as it is done in the single objective case with the incumbent solution and the optimal solution of the relaxed problem of the intermediate nodes of the branch and bound tree). In this way we have an acceleration of the process as several intermediate nodes can be fathomed from LPOS of previous cores' subproblems. However, in large problems with many LPOS, it is advisable to keep the size of the incumbent list in reasonable limits in order to reduce the number of comparisons at every node. This is accomplished with a filtering procedure of the incumbent list in order to keep only a fixed number of them independently of the total number of potential POS. The filtering procedure is actually the forward filtering technique as proposed by Steuer (1989, p. 314) that from a set of vectors provides a subset of the *f* more representative ones. The operation of the algorithm will be illustrated with the educational example of the next section.

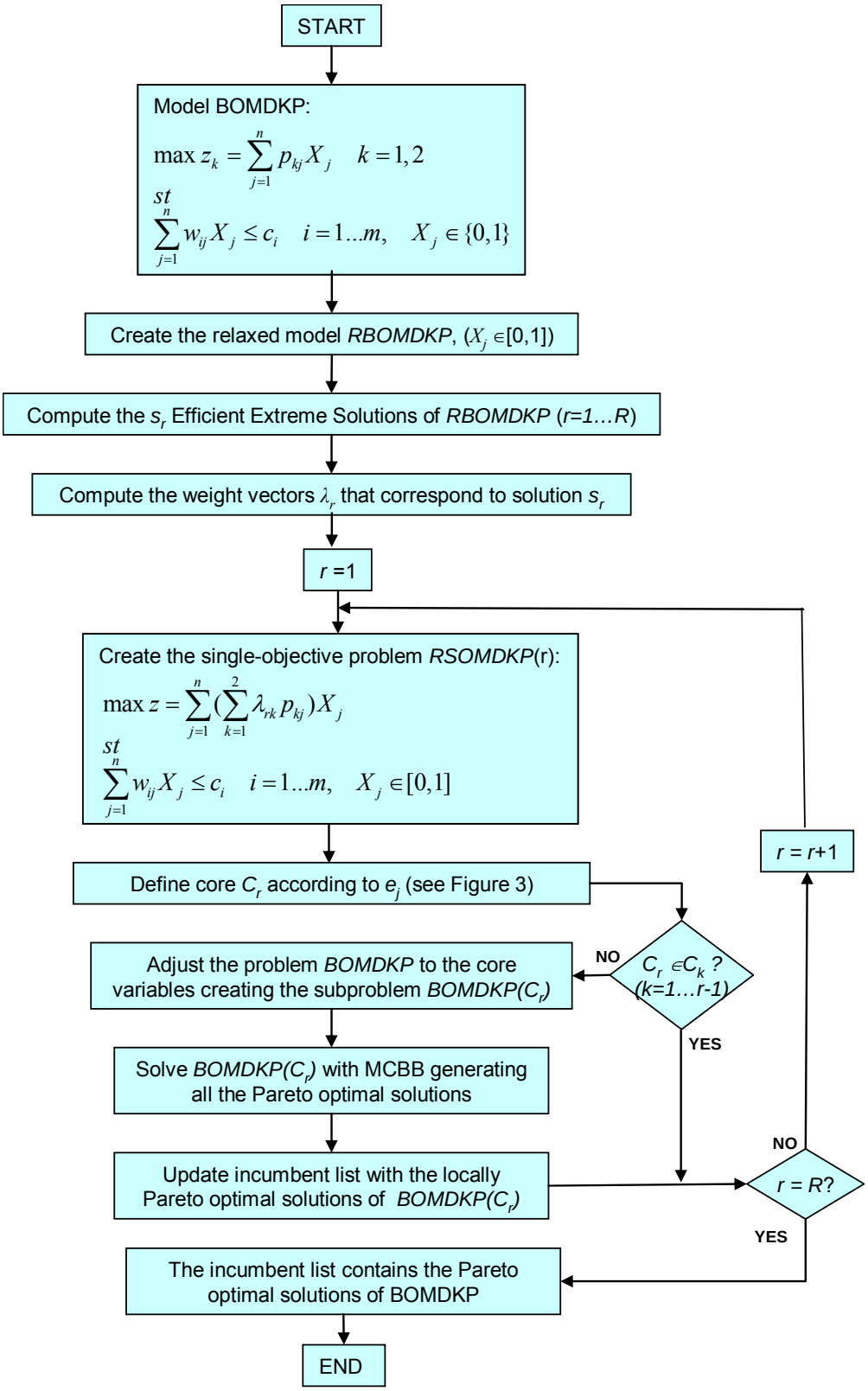

Figure 2: Flowchart of the mcbb-core algorithm

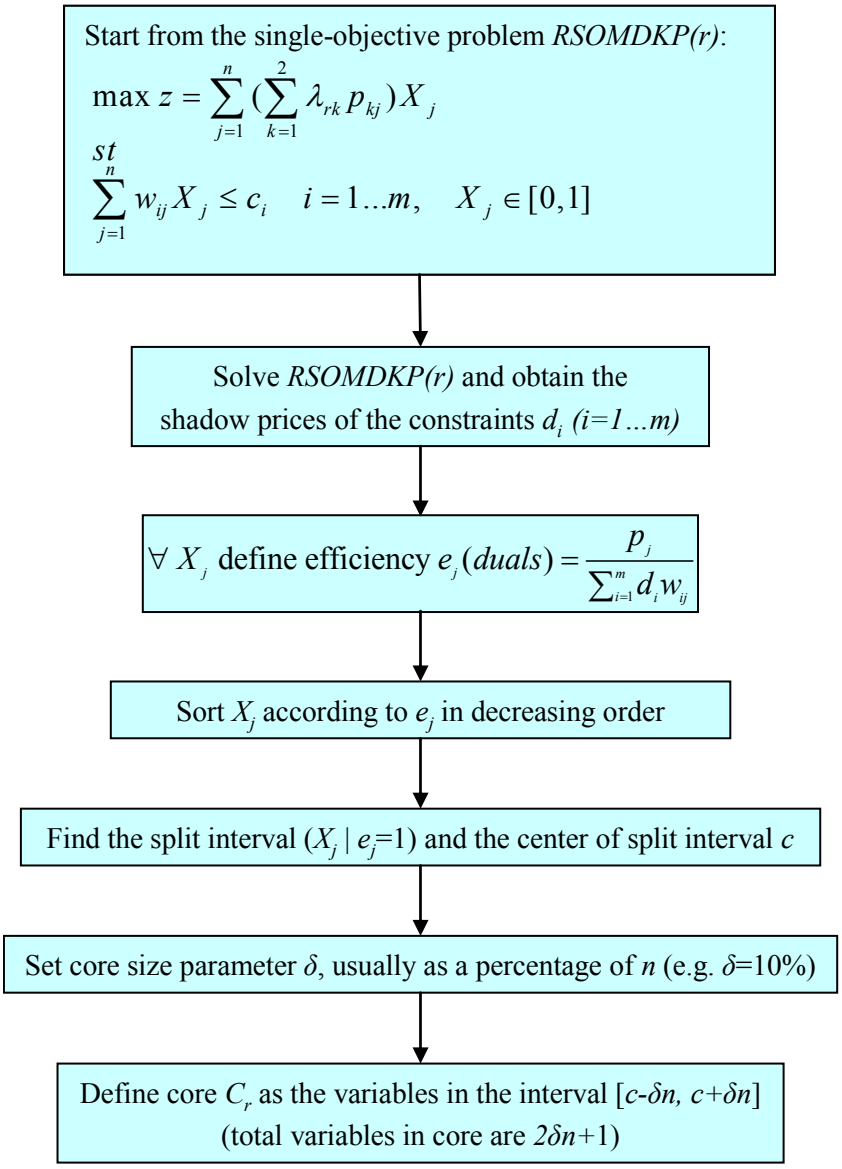

Figure 3: The flowchart for the definition of core

4. Illustrative example

We will use an example and a step by step procedure to illustrate the method. Assume that we have the following BOMDKP problem with m=4 and n=10 (taken from Gomes da Silva et al., 2004, pp. 190).

Problem P410

$$\max\ z_1 = 1x_1 + 87x_2 + 28x_3 + 32x_4 + 38x_5 + 9x_6 + 8x_7 + 6x_8 + 92x_9 + 78x_{10}$$
$$\max\ z_2 = 4x_1 + 21x_2 + 68x_3 + 17x_4 + 43x_5 + 48x_6 + 85x_7 + 30x_8 + 37x_9 + 33x_{10}$$
st
$$70x_1 + 85x_2 + 72x_3 + 31x_4 + 17x_5 + 33x_6 + 47x_7 + 25x_8 + 83x_9 + 28x_{10} \le 246$$
$$49x_1 + 15x_2 + 88x_3 + 29x_4 + 78x_5 + 98x_6 + 50x_7 + 89x_8 + 83x_9 + 3x_{10} \le 291$$
$$15x_1 + 15x_2 + 51x_3 + 3x_4 + 60x_5 + 1x_6 + 78x_7 + 66x_8 + 78x_9 + 71x_{10} \le 219$$
$$56x_1 + 21x_2 + 69x_3 + 60x_4 + 96x_5 + 65x_6 + 100x_7 + 25x_8 + 68x_9 + 30x_{10} \le 295$$
$$x_j \in \{0,1\},\ j = 1,...,10$$

The first step is to solve the relaxed MOLP problem with $x_j \in [0,1]$, which is denoted as R(P410). The problem is solved with the algorithm EFFTREE which provides 11 Efficient Extreme Solutions (EES), the criteria values of which are shown in Table 1 while the corresponding Pareto front is depicted in Figure 4.

Then, according to step 2, we calculate the weight intervals that correspond to every EES. The intervals of these weight coefficients can be obtained from the corresponding simplex tableau (see Steuer, 1989 page 125). The results regarding λ_2 (λ_1=1-λ_2 in the bi-objective case) are shown in Table 1.

Table 1: Efficient extreme solutions of R(P410) and the associated λ_2 intervals.

	Efficient extreme solutions										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
z1	323.0	320.1	317.8	303.5	300.3	264.6	215.1	213.8	192.9	191.5	171.2
z2	151.4	162.9	166.8	184.0	187.3	214.0	245.6	246.3	255.5	255.6	255.7
right edge of λ_2 interval	0.202	0.371	0.453	0.498	0.572	0.610	0.662	0.694	0.913	0.998	1.000

Subsequently we start the loop according to Step 3 that will be repeated 11 times one for each EES. According to step 3a we formulate the single objective LP problems with objective function the appropriate linear combination of the objective functions. For example the objective function of the first problem is:

$$\max z = 0.798\ z_1 + 0.202\ z_2$$
$$= 1.6\ x_1 + 73.7\ x_2 + 36.1\ x_3 + 29\ x_4 + 39\ x_5 + 16.9\ x_6 + 23.6\ x_7 + 10.8\ x_8 + 80.9\ x_9 + 68.9\ x_{10}$$

Solving the LP relaxation (which means $x_j \in [0,1]$) of the above problem we obtain the shadow prices (dual values) of the 4 constraints denoted as d_i. In the present case we have d_1=0.376, d_2=0, d_3=0.535 and d_4=0. Subsequently we form the efficiencies of all the decision variables according to Puchinger's (2006) terminology as expressed in equation (2). For example in the case of x_2 we have:

$$e_2 = \frac{p_2}{\sum_{i=1}^{4} d_i w_{i2}} = \frac{73.7}{(0.376 \times 85 + 0 \times 15 + 0.535 \times 15 + 0 \times 21)} = 1.84$$

Similarly we calculate the efficiencies of all the decision variables and then we rank them in decreasing order taking the results of Table 2.

Table 2: Sorting of variables according to their dual efficiencies in subproblem 1

Rank	1	2	3	4	5	6	7	8	9	10
variable	x_4	x_2	x_{10}	x_9	x_5	x_6	x_3	x_7	x_8	x_1
e_j	2.30	1.84	1.51	1.19	1	1	0.59	0.26	0.19	0.04

The split interval consists of variables (items in knapsack terminology) x_5 and x_6 that have efficiency equal to 1. As it is an even number of items, we take as the center of the split interval the variable x_6. The core is then defined as a set of variables around x_6 . Assuming δ=0.2 we consider $\delta \times n$ variables on the left of the center (which means variables x_5 and x_9) and $\delta \times n$ variables on the right of the center (which means variables x_3 and x_7. Consequently the core contains 5 variables, namely x_9, x_5, x_6, x_3, x_7.

After the calculation of the core variables for the first subproblem we formulate the new model *P410(c1)* that contains only the 5 core variables. Namely, we calculate the fixed part of the objective function that corresponds to the non-core variables and then we calculate the updated RHS of the 4 constraints. The fixed part of the objective functions is calculated according to equation (3) and they are: Z_{1F} $=p_{1,4}+p_{1,2}+p_{1,10}$ =32+87+78=197 and similarly Z_{2F}=17+21+33=71.

$$z_{kF} = \sum_{j \notin lcore} p_{kj} \tag{6}$$

The RHS of the 4 constraints are updated according to equation (4). For example, for the first constraint we have: c'_1= 246-31-85-28 = 102.

After the adaptation of the model to the core variables, we solve *P410(c1)* with MCBB and we obtain 3 Locally Pareto Optimal Solutions (LPOS) that are stored in the incumbent list, which is accordingly updated. Then, we repeat the above steps for the remaining 10 EES of the relaxed problem RBOMDKP. The results are depicted graphically in Figure 4. The complete solution is calculated applying the MCBB method directly to the initial problem with the 10 variables. The approximation with δ=0.2 is reveal that 6 out of the 7 Pareto optimal solutions are discovered.

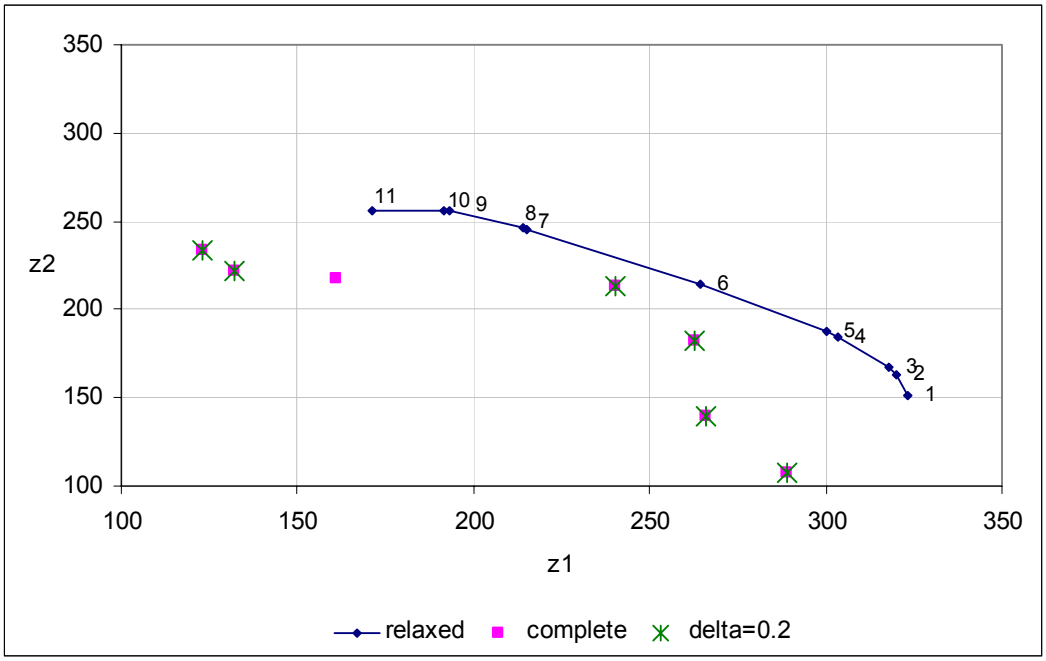

Figure 4: The Pareto front of the relaxed, complete and the approximation with δ=0.2

The sorting results of the variables according to their efficiencies are shown in Table 3 for every EES. The shaded cells are the cores of each subproblem (each core contains 5 variables).

Table 3: Sorting of variables and cores for the 11 efficient extreme solutions

	Rank									
	1	2	3	4	5	6	7	8	9	10
# 1	x_4	x_2	x_{10}	x_9	x_5	x_6	x_3	x_7	x_8	x_1
# 2	x_{10}	x_4	x_5	x_2	x_9	x_6	x_3	x_7	x_8	x_1
# 3	x_{10}	x_5	x_4	x_9	x_2	x_6	x_7	x_3	x_8	x_1
# 4	x_{10}	x_5	x_6	x_9	x_2	x_4	x_7	x_3	x_8	x_1
# 5	x_{10}	x_5	x_6	x_2	x_9	x_7	x_4	x_3	x_8	x_1
# 6	x_{10}	x_2	x_6	x_5	x_9	x_7	x_3	x_4	x_8	x_1
# 7	x_{10}	x_6	x_2	x_7	x_3	x_5	x_9	x_4	x_8	x_1
# 8	x_6	x_{10}	x_2	x_7	x_3	x_5	x_4	x_9	x_8	x_1
# 9	x_6	x_7	x_3	x_{10}	x_5	x_2	x_4	x_9	x_8	x_1
# 10	x_6	x_3	x_7	x_5	x_{10}	x_2	x_4	x_8	x_9	x_1
# 11	x_6	x_3	x_7	x_5	x_{10}	x_8	x_2	x_4	x_9	x_1

The way of calculation of the Pareto optimal solutions using the core concept is better illustrated in Figure 5. In this graph we show from which core all the Pareto optimal solutions are generated.

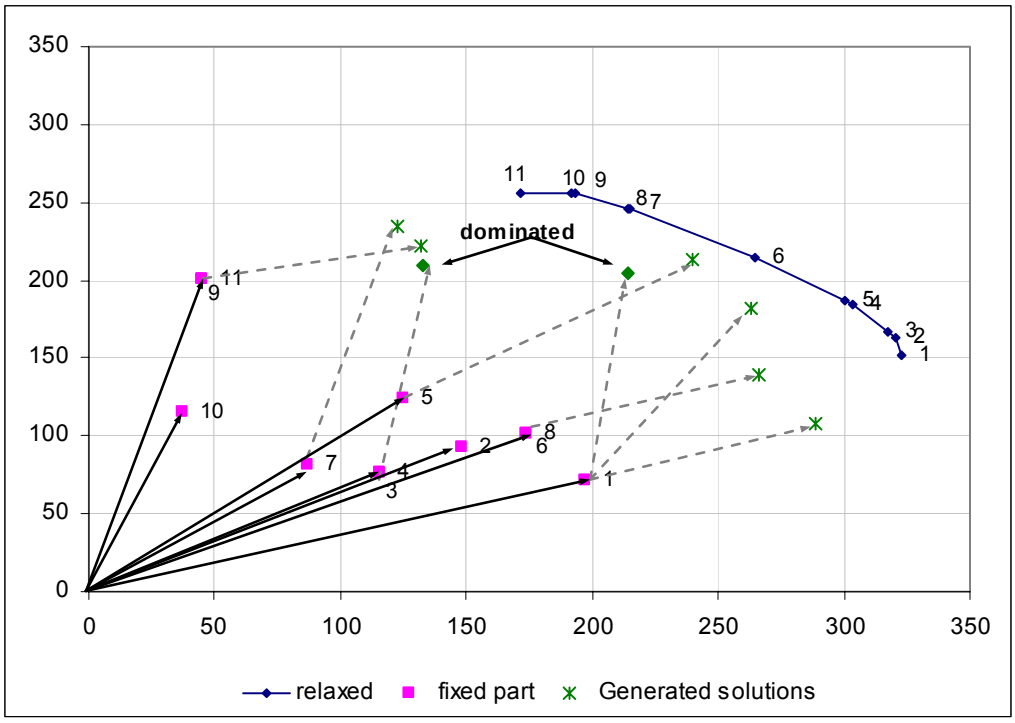

Figure 5: Schematic representation of the generation of the Pareto optimal solutions of the approximation with δ=2.

Note that for the efficient extreme points 4, 8 and 11 we avoid the calculation of the LPOS as their core is identical to previously computed 3, 6 and 9. The Pareto optimal solutions from each one of the core subproblems are considered to have a fixed part and a variable part. The fixed part is the (z_{1F}, z_{2F}) vector and the variable part represents the different Pareto optimal solutions obtained from each core. Observe that there are cores that have the same fixed part (3and 4, 6 and 8, 9 and 11). There are core problems that produce no Pareto optimal solutions due to the fathoming condition (2 and 10) and there are LPOS that are dominated by the LPOS of another's core subproblem (they are indicated with a diamond in Figure 5). Specifically, these are the 3rd LPOS from core 1 (dominated by the LPOS generated by core 5) and the LPOS from core 3 (dominated also by the core's 5 LPOS). It must be noted that if we increase the core size parameter δ to 0.35 then all the 7 Pareto optimal solutions are generated.

5. Adjusted core

In the great majority of MOLP problems the dispersion of the EES in the Pareto front is not uniform (see e.g. Figure 4). In several examples we observed that the approximation of the Pareto set was poor in the areas where the EES were away one from the other. On the contrary, where the EES were close one to the other the approximation was very rich. In Figure 6 the Pareto sets of a BOMDKP with n=40 and m=3 are depicted. The example was taken from Laumanns et al. 2006 (the original example had 3 objective functions and for our purposes we delete the third). The relaxed MOLP has 26 efficient extreme solutions and the MOMDKP has 34 Pareto optimal solutions. The complete Pareto set was computed with the conventional MCBB (running for 40 variables) in 1'36" (in a Intel Core 2 Duo 2.66 GHz machine).Using the core issue with δ=0.1 we obtain the 21 solutions shown in Figure 3 in less than 1 second. If we expand

the core setting $\delta = 0.2$ (which is translated to 17 variables in the core) we obtain all the 34 Pareto optimal solutions in 12" which is a significant reduction in solution time.

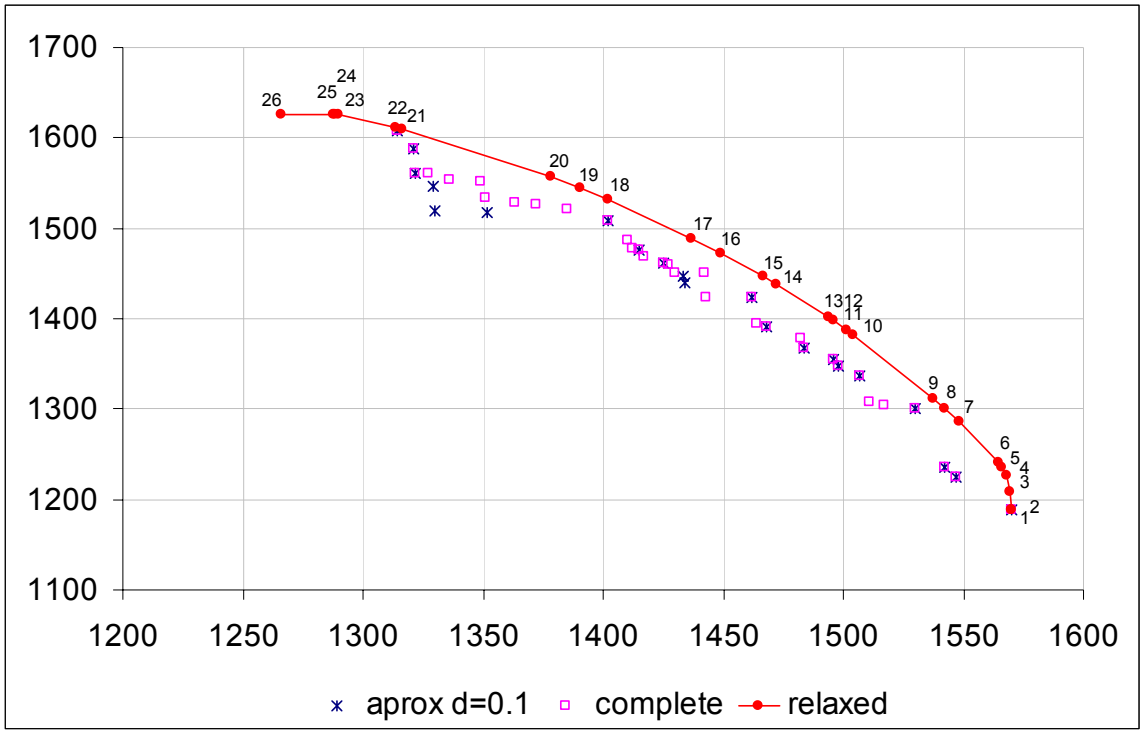

Figure 6: Results from a problem with 40 variables and 3 constraints (n=40, m=3)

Observing the results it can be seen that in the sparse areas of the relaxed Pareto front the approximation is poor (see e.g. between points 9-10, 17-18 and 20-21). This is rational as we solve one subproblem for every EES, performing so a more thorough search in the dense areas. So we conclude that in the sparse areas we need to enhance the representation by using a greater core in the relevant subproblems. On the contrary, in the dense areas a smaller core would suffice as many subproblems are going to be solved, decreasing the probability of losing some Pareto optimal solutions.

A rational idea is to adjust the size of the core according to the Euclidean distance between one EES and its next (normalized to the total length of the Pareto front which is actually the sum of the *R-1* inter-point distances). The size of the core is adjusted by properly adjusting the δ parameter. The formula for the adjusted δ that we assumed is the following:

$$\delta_i = \delta_0 + \delta_1 \times \frac{l_i}{l_{avg}} \qquad \text{for } i = 1 \ldots R \tag{7}$$

Where l_i is the Euclidean distance between EES_i and EES_{i+1}, l_{avg} is the average length (=total length of Pareto front/(R-1)), δ_0 is the intercept that express the minimum value for δ and δ_1 is the slope that indicates how fast δ_i is increasing with (l_i/l_{avg}). In order to avoid undesired large cores, we also bound the value of δ_i by a parameter δ_{max} so that the complete formula becomes:

$$\delta_i = \min(\delta_{\max}, \delta_0 + \delta_1 \times \frac{l_i}{l_{avg}}) \qquad \text{for } i = 1 \ldots R \tag{8}$$

It must be noted that with the adjusted delta, the solution time dropped from 12" to 2.47" and produce all the 34 Pareto solutions providing the complete solution. The number of core variables for each one of the 26 EES is shown in Table 4:

Table 4: Core elements for the 26 efficient extreme solutions

EES	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Core #	5	9	9	7	7	13	7	7	19	7	7	5	13	7	11	9	15	9	9	21	5	11	5	5	9	9

By examining Figure 6 one can observe that in the areas where we have accumulation of efficient extreme points the δ parameter is low and increases for those EES that are away from their next (e.g. see EES 6, 9, 13, 17, 20 with cores 13, 19, 13, 15 and 21 variables respectively).

The main conclusion is that using an adaptive core we can achieve computational economy without losing goodness of fit, with the latter measured as the degree of approximation to the complete Pareto set.

6. Computational experiments

In the computational experiments we used available benchmarks where the complete set of the Pareto optimal solutions is known or can be computed with the MCBB method. Initially, we use small benchmarks that are taken from Chu and Beasley's benchmark library that are properly modified. Subsequently, we design a computational experiment with instances derived also from Chu and Beasley in order to examine the effect of the core size on the algorithm effectiveness. The size of the examined problems was chosen so that MCBB can produce the complete Pareto set in reasonable time. Finally, we test the algorithm in larger problems with known complete Pareto set as provided by Laumanns et al. The complete Pareto sets have been computed with the adaptive ε-constraint method (Laumanns, et al. 2006). All the runs were performed in an Intel Core 2 Duo T7250 at 2 Ghz.

6.1 Comparison with conventional MCBB in small problems

The first set of models are derived from the Chu and Beasley knapsack problems library found in http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html. The only required intervention is that a second objective function is added. The coefficients of the 2nd objective function are taken randomly from an appropriate uniform distribution. The number of variables (n) varies from 6 to 50 and the number of constraints (m) is 5 and 10. Specifically, the sizes of the 7 problems, in $n \times m$ format, are 6×10, 10×10, 15×10, 20×10, 28×10, 39×5 and 50×5.

First, we solve the 7 problems with the original MCBB method generating the complete set of the Pareto optimal solutions (POS). Then we solve the 7 problems with the proposed method (COREMCBB) using the core size parameter δ=0.25, which means that half of the original variables are actually used in each core sub-problem. The required efficient extreme solutions of the corresponding relaxed problems are also computed. The relevant Pareto sets for the last six models are depicted in Figure 7 and the results are shown in Table 5.

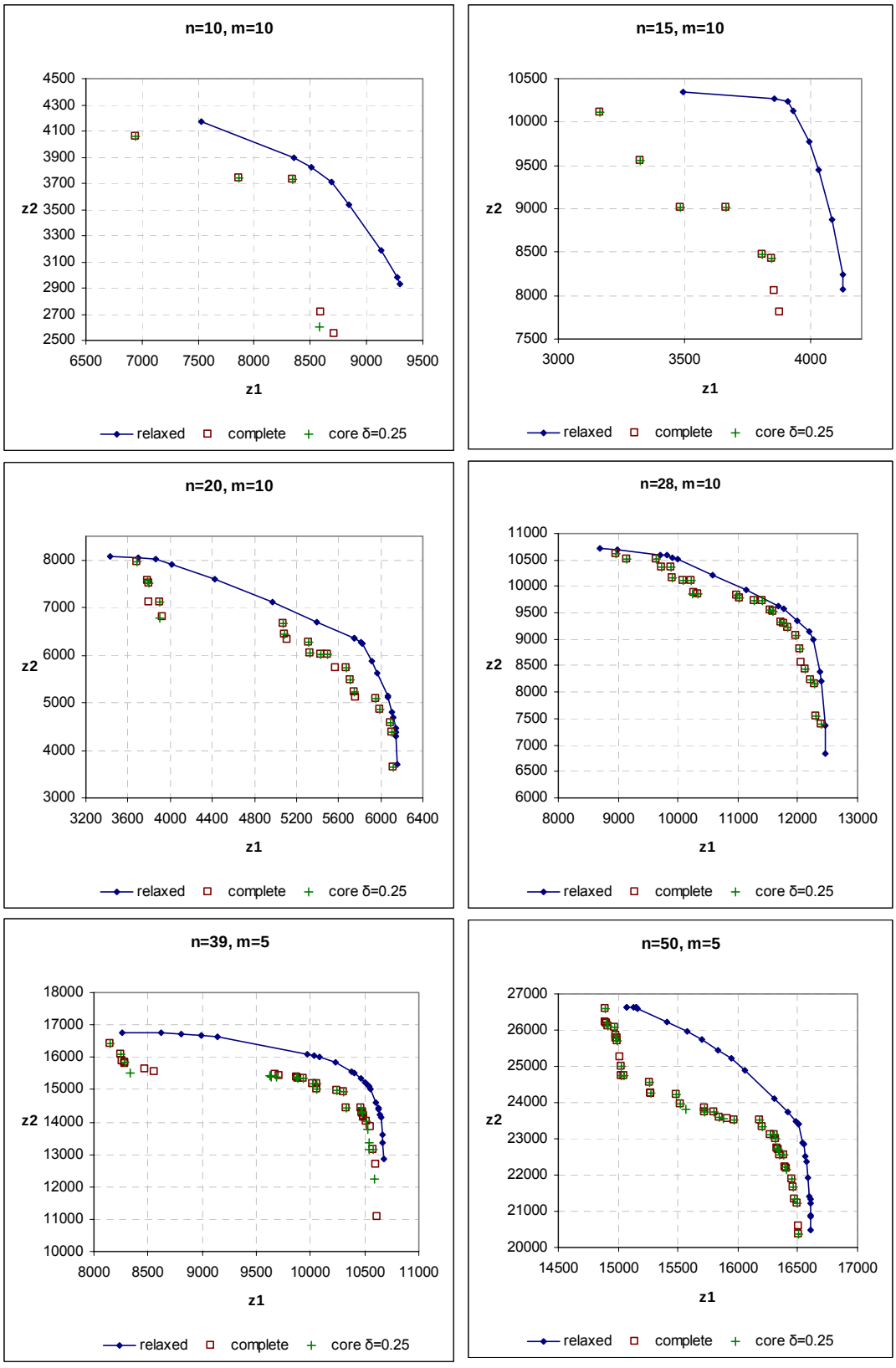

Figure 7: Pareto sets of the 6 benchmarks from Chu&Beasley

Table 5: Results for the 7 benchmarks

	number of Pareto optimal solutions				time (sec)		time fraction %
	relaxed*	complete	δ=0.25	coverage	complete	δ=0.25	
6×10	7	2	2	100%	0.1	0.1	-
10×10	8	5	4	60%	0.1	0.1	-
15×10	9	11	7	64%	0.4	0.1	25.0%
20×10	21	23	19	78%	5.0	1.0	20.0%
28×10	17	27	26	93%	5.3	3.2	60.4%
39×5	24	29	26	59%	30.1	6.2	20.6%
50×5	28	45	38	73%	24264	64.5	0.3%

* For the relaxed problem, the figures refer to the efficient extreme solutions

The coverage in Table 5 is defined as the percentage of the POS from the complete Pareto set (generated with the MCBB) that are found also with the approximate method, COREMCBB. The time fraction expresses the computational economy and refers to the quotient with nominator the COREMCBB solution time and denominator the MCBB solution time. We can observe that the beneficial effect of COREMCBB is more apparent in the instances with many decision variables, where the computational time explodes. In the case with n=50 the computational economy is huge, while the coverage is substantial (73%). We will take a closer look to this case (problems with n=50 and m=5) in the next paragraph.

6.2 Effect of the core size on the effectiveness of the algorithm

Ten benchmark problems with n=50 and m=5 were generated by sub-sampling from a problem with n=100 and m=5. In the sub-sampling process we randomly choose 50 out of the 100 columns (variables) of the model. By doing this 10 times we create 10 problems with n=50 and m=5. The right hand side (RHS) of the constraints was defined appropriately, so that the tightness ratio in all constraints is 0.5.

The core size parameter δ was taken constant in four cases (δ=0.1, 0.15, 0.2, 0.25) and adjusted in two cases (δ_0=0.1, δ_1=0.1 and δ_{max}=0.2 in the first case and δ_0=0.1, δ_1=0.1 and δ_{max}=0.25 in the second case). The results are shown in Table 6. It is observed in the results that although the number of POS of the complete Pareto set doesn't vary too much, the computation time may differ dramatically (from 2443 to 84193 seconds which is more than 35 times more!). As it was expected, the increase in core size (expressed by the δ parameter) results in better representation of the Pareto set (reflected in "coverage"), but with higher computation times. There is always a trade-off between coverage and computation time. It was also observed that the two cases with adjusted core do not clearly outrank the cases with the fixed core and more computational experiments are needed to draw more meaningful conclusions. The average values for coverage and time fraction are graphically depicted in Figure 8.

Table 6: Results from 10 instances of the n=50, m=5 case

		relaxed*	complete	δ=0.1	δ=0.15	δ=0.2	δ=0.25	δ-adj (0.1,0.1,0.2)	δ-adj (0.1,0.1,0.25)
number of Pareto optimal solutions	Min	27	25	9	21	23	25	22	23
	Max	52	41	27	38	39	40	39	40
	Avg	41.3	34.3	19.4	29.9	31.8	33.2	31.0	32.4
	Sd	8.2	5.8	5.0	5.7	5.2	5.7	5.2	5.5
coverage	Min	-	-	12.0%	48.0%	69.0%	83.0%	61.0%	72.0%
	Max	-	-	45.0%	82.0%	96.0%	100.0%	96.0%	100.0%
	Avg	-	-	30.1%	66.2%	85.2%	94.5%	81.4%	89.3%
	Sd	-	-	10.9%	13.2%	9.5%	5.5%	11.9%	8.3%
time (sec)	Min	0.4	2443	1.0	10.0	45	132	24	41
	Max	0.9	84193	2.0	19.0	109	563	77	318
	Avg	0.6	17238	1.8	14.4	65	247	38	120
	Sd	0.18	23949	0.4	3.0	23	143	17	85
time fraction	Min	-	-	-	0.02%	0.11%	0.51%	0.06%	0.21%
	Max	-	-	-	0.78%	4.46%	23.05%	3.15%	13.02%
	Avg	-	-	-	0.19%	0.94%	3.99%	0.60%	2.09%
	Sd	-	-	-	0.21%	1.28%	6.78%	0.92%	3.88%

* For the relaxed problem, the figures refer to the efficient extreme solutions

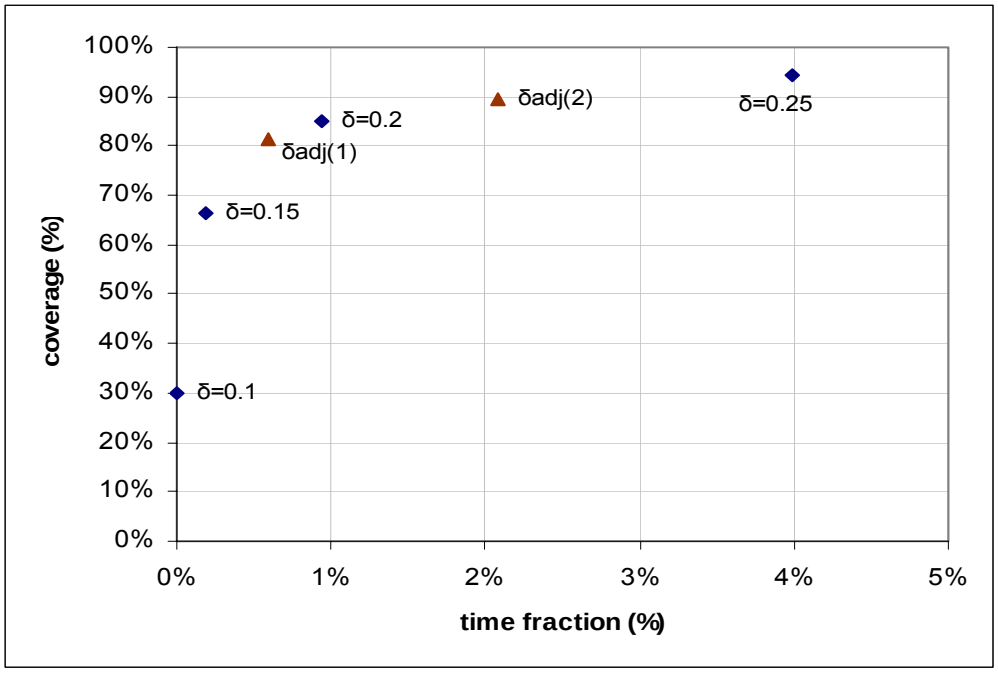

Figure 8: Average values for coverage vs time fraction for different core sizes from 10 examples with n=50, m=5 (δadj(1) is the case of δ_{max}=0.2 and δadj(2) the case of δ_{max}=0.25)

From Figure 8 it is observed that exploiting the core issue, we can achieve in less than 1% of the computational time more than 80% coverage. However the trade-off curve between time fraction and coverage becomes more horizontal as we move to greater coverage values. Specifically, we can achieve a substantial increase in coverage from δ=0.1 to δ=0.2 with a low sacrifice in computational time. However, a further increase in coverage costs much more in computational time.

6.3 Larger problems

We also test COREMCBB in larger problems with known complete Pareto sets that were found in http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite. The complete Pareto sets were probably generated using the adaptive ε-constraint method (Laumanns et al., 2006). The first problem has n=100 and m=2. We test four fixed core sizes, namely δ=0.05, 0.075, 0.1, 0.125 which means 11, 16, 21, 26 core items respectively and one adjusted core size. The results are shown in Table 7.

Table 7: Results for the n=100, m=2 case

	relaxed*	complete	δ=0.05	δ=0.075	δ=0.1	δ=0.125	Δ-adj (0.05, 0.025, 0.125)
number of POS	53	121	87	114	118	120	118
coverage	-	-	57%	88%	92%	94%	89%
time (sec)	-	-	4	39	231	1246	246

The second problem has n=250 and m=2. We test four fixed core sizes, namely δ=0.02, 0.03, 0.4, 0.5 which means 11, 16, 21, 26 core items respectively and one adjusted core size. The results are shown in Table 8.

Table 8: Results for the n=250, m=2 case

	relaxed*	complete	δ=0.02	δ=0.03	δ=0.04	δ=0.05	δ-adj (0.02,0.01,0.05)
Number of POS	154	567	243	457	517	555	485
Coverage	-	-	26%	63%	81%	93%	71%
Time (sec)	-	-	13	183	1798	77626	4325

7. Conclusions and discussion

In the current work we define the core concept for the multi-objective multidimensional knapsack problem and we develop a method based on the core concept for the bi-objective case. A cornerstone of the method is the MCBB algorithm properly adjusted to the pure integer case. The results from the "divide and conquer" strategy are very promising regarding the quality of the approximation and the solution time.

The points for future work are the following:

- Expand the application of the method to more than two objectives with the corresponding computational results.
- Upgrade the approximate method of this document to be an exact method on its own right. A first approach is to exploit the obtained POS and search exhaustively for other POS not found by the COREMCBB method using multiple IP models. The obtained results can be used to effectively

expand the already found cores and repeat the COREMCBB approach. This iterative process eventually discovers all the POS of the BOMDKP. The generation of the complete Pareto set in such problems will be very useful for benchmarking the widely used nowadays multiobjective metaheuristics.

- The algorithm is very suitable for parallelization as the core subproblems C_r can be assigned to different CPUs of the same machine.

References

Gomes da Silva, C., Climaco, J., Figueira, J., 2004. A scatter search method for the bi-criteria multi-dimensional {0,1}-knapsack problem using surrogate relaxation, Journal of Mathematical Modelling and Algorithms 3, 183-208.

Gomes da Silva, C., Climaco, J., Figueira, J., 2008. Core problems in bi-criteria {0, 1}-knapsack problems, Computers & Operations Research 35, 2292-2306.

Fréville A., 2004. The multidimensional 0-1 knapsack problem: An overview. European Journal of Operational Research 155, 1-21.

Kellerer, H., Pferschy U., Pisinger, D., 2004. Knapsack Problems. Springer. Berlin

Laumanns, M., Thiele, L., Zitzler, E., 2006. An efficient, adaptive parameter variation scheme for Metaheuristics based on the epsilon-constraint method. European Journal of Operational Research 169, 932-942.

Martello, S., Toth., P., 1990. Knapsack problems - algorithms and computer implementations, Wiley, New York.

Mavrotas, G., 2000, Multiple Objective Programming under Uncertainty: Development of a Decision Support System and Implementation in Energy Planning. Ph.D. Thesis, National Technical University of Athens, Athens.

Mavrotas, G., Diakoulaki, D., 1998. A Branch and Bound Algorithm for Mixed Zero-One Multiple Objective Linear Programming. European Journal of Operational Research 107, 530-541.

Mavrotas, G., Diakoulaki, D., 2005. Multi-criteria branch & bound: a vector maximization algorithm for mixed 0-1 multiple objective linear programming. Applied Mathematics & Computation 171, 53-71.

Pisinger, D., 1995. Algorithms for Knapsack problems. Ph.D. Thesis.Dept. of Computer Science, University of Copenhagen.

Puchinger, J., Raidl, G.R., Pferschy, U., 2006. The core concept for the multidimensional knapsack problem, in: Gottileb, J. and Raidl, G.R. (eds.), LNCS 3906, pp.195-208.

Steuer, R.E. 1989. Multiple Criteria Optimization-Theory, Computation and Application (2nd ed). Krieger, Malabar FL.

Steuer, R.E., 1995. The ADBASE Multiple Objective Linear Programming Package. In: Gu J, Chen G, Wei Q, Wang S (ed) Multiple Criteria Decision Making. Windsor,England, pp. 1-6