AUGMECON2: A novel version of the ε-constraint method for finding the exact Pareto set in Multi-Objective Integer Programming problems

George Mavrotas, Kostas Florios

Laboratory of Industrial and Energy Economics, School of Chemical Engineering,
National Technical University of Athens, Zographou Campus, Athens 15780, Greece, Tel: +30 210-7723202,
fax: +30 210 7723155, e-mail: mavrotas@chemeng.ntua.gr

Abstract: Generation (or a posteriori) methods in Multi-Objective Mathematical Programming (MOMP) is the most computationally demanding category among the MOMP approaches. Due to the dramatic increase in computational speed and the improvement of Mathematical Programming algorithms the generation methods become all the more attractive among today's decision makers. In the current paper we present the generation method AUGMECON2 which is an improvement of our development, AUGMECON. Although AUGMECON2 is a general purpose method, we will demonstrate that AUGMECON2 is especially suitable for Multi-Objective Integer Programming (MOIP) problems. Specifically, AUGMECON2 is capable of producing the exact Pareto set in MOIP problems by appropriately tuning its running parameters. In this context, we compare the previous and the new version in a series of new and old benchmarks found in the literature. We also compare AUGMECON2's performance in the generation of the exact Pareto sets with established methods and algorithms based on a specific MOIP problem (knapsack) and on published results.

Keywords: Multi-Objective Programming, ε-constraint method, exact Pareto set

1. Introduction

The rapid improvement in computer performance, software and algorithms transformed the almost unsolvable calculation problems of previous decades into trivial tasks. This is especially true for Mathematical Programming where problems with thousand of variables and constraints can be solved in seconds or minutes. It is well known that the Multiple Objective Mathematical Programming refers to the solution of Mathematical Programming problems with more than one objective functions. Given that usually there is no unique optimal solution (optimizing simultaneously all the objective functions), the aim is to find the most preferred among the Pareto optimal solutions [1]. Therefore, MOMP methods have to combine optimization with decision support.

In the late seventies Multiple Objective Mathematical Programming methods were classified by Hwang and Masud into three classes according to the phase in which the decision maker was involved in the decision making process [2]: The a priori methods, the interactive methods and the a posteriori or generation methods. The latter class was somehow neglected at this time as it was the most computationally demanding and only small problems could be addressed mostly with academic software.

Nowadays, with the vast improvement in computer power the generation approaches become all the more popular as they have some significant advantages. The solution process can be divided into two separate phases: Phase one is the generation of the Pareto optimal solutions (all or a subset of them). Phase two is the subsequent involvement of the decision maker when all the information is on the

table. No re-optimizations and further interaction are needed (especially important whenever the DM is hardly available) and the fact that none of the potential solutions has been left undiscovered, reinforces the DM's confidence on the final decision.

AUGMECON is a generation method introduced by Mavrotas in [3]. It is an improvement of the original ε-constraint method which is along with the weighting method one of the two most popular methods for generating representations of the Pareto front. As it is described in [3], the ε-constraint method has certain advantages in relation to the weighting method especially in the presence of discrete variables (Mixed Integer or Pure Integer problems). In the current work we are going one step further, introducing AUGMECON2 an improvement of AUGMECON that exploits the information from the slack variables in every iteration. The improvements regard the reduction in computation time as many redundant iterations are avoided.

These improvements are more effective when the problem contains discrete variables and the feasible region is non-convex. AUGMECON2 proved to be very efficient in Multi-Objective Integer Programming (MOIP) problems where the Pareto set is finite and countable. The AUGMECON2 can be used in order to effectively produce approximations of the Pareto set. By controlling the number of grid points, we control the density of the approximation. However, its major advantage is that in MOIP problems we can adjust the method in order to produce all the Pareto optimal solutions (exact Pareto set).

This is extremely important as the optimization community is interested in methods producing the exact Pareto set in Multi-Objective Combinatorial Optimization (MOCO) problems (see e.g. [4-6]). Various methods have been proposed, either generic or specific (for specific MOCO problems like e.g. the knapsack problem) that are able to find all the Pareto optimal solutions in MOCO problems. This kind of problems can be formulated as mathematical programming problems and more specifically as MOIP problems (usually with only 0-1 variables).

The rest of the paper is organized as follows: In Section 2 the novel parts of the AUGMECON2 method are described and the application of the proposed method in the generation of the exact Pareto front in Multiobjective Integer Programming problems is illustrated. In Section 3 the computational experiment for the evaluation of the proposed method in MOIP test problems is described and the results of the comparison of AUGMECON2 with other methods (including its older version) in a variety of new and existing in the literature test problems are discussed in Section 4. Finally in Section 5 the basic concluding remarks are discussed.

2. Methodological part

2.1 The improved version of the augmented ε-constraint (AUGMECON2)

We start this section by briefly describing the original version of the augmented ε-constraint method, named AUGMECON [3] for the integrality of the paper. AUGMECON enhances the conventional ε-constraint method for generating the Pareto optimal solutions in Multi-Objective Mathematical Programming problems. It is well known that the ε-constraint has certain advantages in relation to the weighting method as described in [3]. AUGMECON addresses some weak points of the conventional ε-constraint, namely, the guarantee of Pareto optimality of the obtained solution in the payoff table as well as in the generation process and the increased solution time for problems with several (more than two) objective functions.

In the original AUGMECON method the problem solved is the following:

$\max (f_1(\mathbf{x}) + eps \times (S_2/r_2 + S_3/r_3 + \ldots + S_p/r_p))$

st

$f_2(\mathbf{x}) - S_2 = e_2$

$f_3(\mathbf{x}) - S_3 = e_3$ (1)

. . .

$f_p(\mathbf{x}) - S_p = e_p$

$\mathbf{x} \in S$ and $S_i \in R^+$

where e_2, e_3,...,e_p are the parameters for the RHS for the specific iteration drawn from the grid points of the objective functions 2,3,...,p. The parameters r_2, r_3,...,r_p are the ranges of the respective objective functions. S_2, S_3,..., S_p are the surplus variables of the respective constraints and $eps \in [10^{-6}, 10^{-3}]$.

In AUGMECON2, the improved version of AUGMECON we slightly modify the objective function as follows:

$$\max (f_1(\mathbf{x}) + eps \times (S_2/r_2 + 10^{-1} \times S_3/r_3 + \ldots + 10^{-(p-2)} \times S_p/r_p))$$

This modification is done in order to perform a kind of lexigographic optimization on the rest of the objective functions if there are any alternative optima. For example, with this formulation the solver will find the optimum for f_1 and then it will try to optimize f_2, then f_3 and so on. With the previous formulation the sequence of optimizations of f_2 to f_p was indifferent, while now we force the sequential optimization of the constrained objective functions (in case of alternative optima).

As it is explained in [3], for each objective function *2...p* we calculate the objective function range. Then we divide the range of the k-th objective function to q_k equal intervals using (q_k-*1*) intermediate equidistant grid points. Thus we have in total (q_k+*1*) grid points that are used to vary parametrically the RHS (e_k) of the k-th objective function. The total number of runs becomes (q_2+*1*) × (q_3+*1*) × . . . × (q_p+*1*). Let r_k be the range of the objective function k (k=2...p). Then the discretization step for this objective function is given as:

$$step_k = r_k/q_k$$

The RHS of the corresponding constraint in the t-th iteration in the specific objective function will be:

$$e_{kt} = fmin_k + t \times step_k$$

where $fmin_k$ is the minimum obtained from the payoff table and *t* the counter for the specific objective function.

In each iteration we check the surplus variable that corresponds to the innermost objective function. In this case it is the objective function with *p*=2. Then we calculate the *bypass coefficient* as:

$$b = int(S_2/step_2)$$

where *int( )* is the function that returns the integer part of a real number. When the surplus variable S_2 is larger than the $step_2$, it is implied that in the next iteration the same solution will be obtained with the only difference being the surplus variable which will have the value S_2-$step_2$. This makes the

iteration redundant and therefore we can bypass it as no new Pareto optimal solution is generated. The bypass coefficient *b* actually indicates how many consecutive iterations we can bypass.

This can be shown with the following example: Assume that we have a three-objective problem with the following payoff table (all objective functions to be maximized):

Table 1. Payoff table

	$f_1(x)$	$f_2(x)$	$f_3(x)$
max $f_1(x)$	980	796	803
max $f_2(x)$	836	876	765
max $f_3(x)$	809	821	905

From the payoff table we have r_2=80 and r_3= 140. We then divide the two ranges in 10 equal intervals with $step_2$=8 και $step_3$=14. The AUGMECON2 process is the following:

For i=0 to 10
  $e_3=765 + i\times14$
  For j=0 to 10
    $e_2=796 + j\times8$
    Solve (1)
  Next j
Next i

The objective function $f_2(x)$ is the innermost loop (j counter). Assume that we are in the 2nd outermost (where i=1) and the 5th innermost iteration (where j=4) where e_3=779 and e_2=828 which are the shaded cells in Table 2.

Table 2. Grid points of the problem

obj. function	counter	grid points										
		0	1	2	3	4	5	6	7	8	9	10
$f_2(x)$	j	796	804	812	820	828	~~836~~	~~844~~	852	860	868	876
$f_3(x)$	i	765	779	793	807	821	835	849	863	877	891	905

After the optimization we obtain S_2=18 and S_3=9, which mean that in this iteration the value for the second objective function is:

$f_2 = e_2 + S_2 = 828+18 = 846$ and $f_3 = e_3 + S_3 = 779+9 = 788$.

We conclude that it is redundant to perform the next two iterations with j=5 and j=6 (strikethrough in table 2) because we will result in the same Pareto optimal solution with f_2=846. The only difference is that the surplus variables will be 10 (=18-8) and 2 (18-2×8) for j=5 and j=6 respectively. Therefore we can bypass these two iterations and go directly from j=4 to j=7 (value for e_2=852 in Table 2). The bypass coefficient b is calculated as $b = int(18/8) = 2$. The flowchart of the new algorithm is shown in Figure 1:

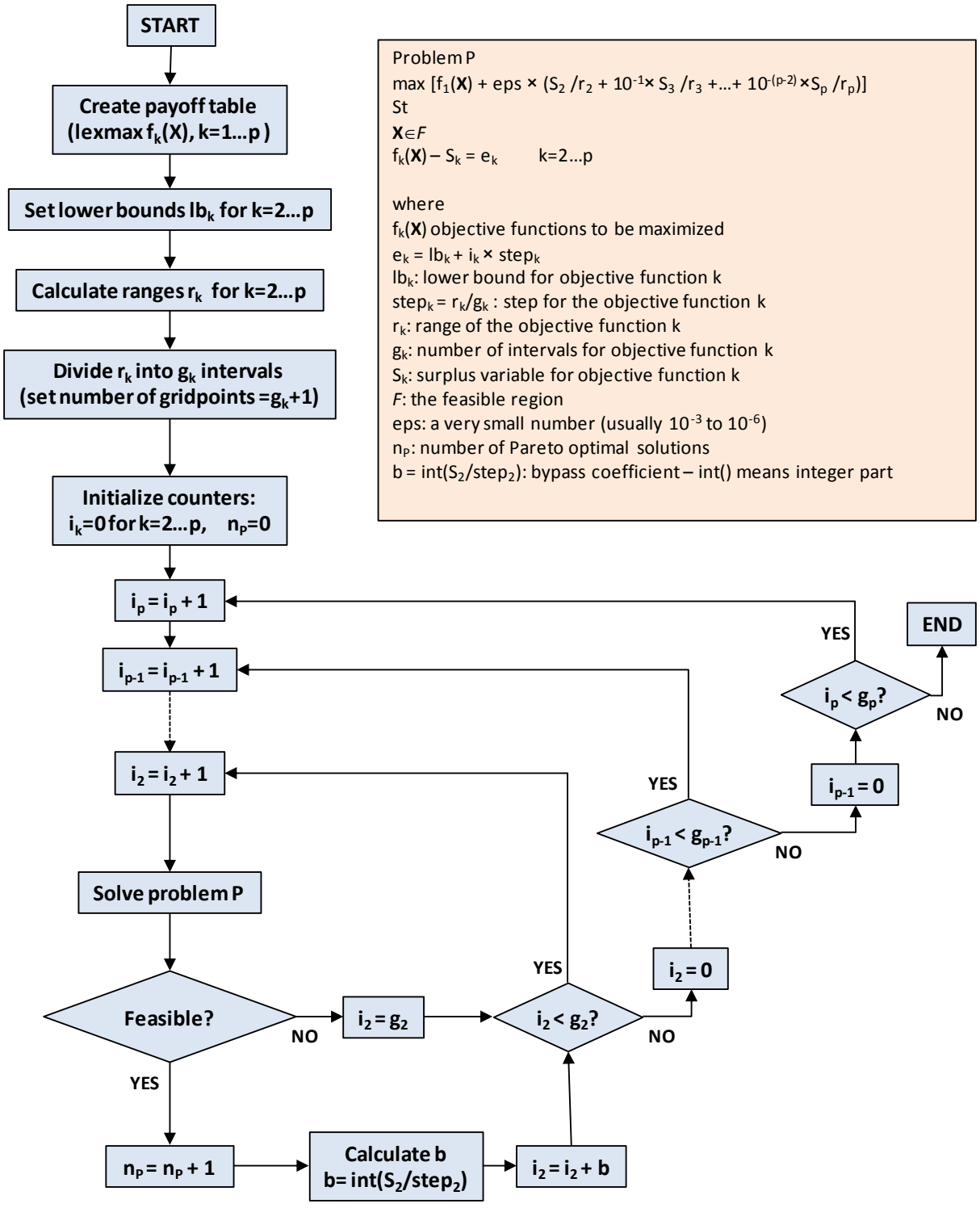

Figure 1. *Flowchart of the AUGMECON2 method*

In this way, using the bypass coefficient to exploit the information from the slack/surplus variables of the constrained objective functions we greatly accelerate the algorithm as we avoid redundant iterations. As we will see in the subsequent section the "jumps" caused in the innermost loop become even more significant when we reduce the step size (increase the grid density). Therefore, we take advantage of this property in order to produce the exact Pareto front in MOIP problems in reasonable

computation time. The GAMS code with some instructions and instances/results of this paper are available in https://sites.google.com/site/kflorios/.

2.2 Calculation of the exact Pareto set in MOIP problems using AUGMECON2

As it was mentioned, the ε-constraint method can be properly used for the generation of the Pareto optimal solutions in MOMP problems with discrete variables. In the case of MOIP and 0-1MOMP problems (MOIP problems with only 0-1 variables as integer variables which are the vast majority of MOIP problems) the ε-constraint method can be used to produce the exact (or complete) Pareto set, which means all the Pareto optimal solutions. As it is well known the size of the Pareto set is finite in MOIP problems and the AUGMECON2 is therefore suitable for generating the exact Pareto set for these problems. The only conditions are the following:

1. The objective function coefficients must be integer
2. The nadir points of the Pareto set must be known

In order to relax the first condition we can easily transform the problem to have integer objective function coefficients by multiplying with the appropriate power of 10. For example if we have one decimal digit in the objective function coefficients, we multiply them with 10, if we have two decimal digits we multiply them with 100. It must be noted that the greater the magnitude of the objective function coefficients the more time consuming is the generation of the exact Pareto set (see e.g. section 3 in the computational experiment). The reason is that the range of the objective functions is inevitably greater.

In order to relax the second condition it is adequate to know a lower bound of the nadir point (for maximization problems). The payoff table in multi-objective programming provides the individual optima of the objective functions (in the diagonal). However, it is well known (see e.g. Isermann and Steuer [7]) that it does not guarantee the calculation of the nadir points except only for problems with just two objective functions. For problems with three and more objective functions the minima of the payoff table's columns do not necessarily provide the nadir point of the problem. Consequently, in order to be sure that we have the true nadir points we cannot rely on the values of the payoff table. We must either calculate them using one of the methods in the literature, or estimate some lower bounds of them. The latter usually results in slightly higher computational effort afterwards, during the AUGMECON2 process. The closer is the lower bound to the nadir point the less is the computational time as the objective function range becomes narrower.

In order to calculate the nadir point several approaches have been proposed in the literature (see e.g. [7-11]). In the case of MOIP problems we find the procedure of Jorge [11] as more suitable to our case. Specifically, we just reverse the direction of optimization of the *p-1* objective functions and optimize them over the integer efficient set, obtaining thus the elements of the nadir vector.

The computational strategy for calculating the exact Pareto set in MOIP problems with integer objective function coefficients is the following (without loss of generality, assume that all the objective functions are to be maximized):

1. Calculate the objective function ranges of the *p-1* objective functions. This means that we have to calculate or estimate lower bounds for the nadir points. If the nadir points are not given (e.g. in the literature for the specific problem), we apply the method of Jorge [11] for calculating them.

2. Assume that the objective function range for the k-th objective function is r_k (integer). We select for each objective function a unity step so that for each one of them the number of grid points is exactly r_k+1
3. We apply AUGMECON2 and obtain the exact Pareto set. The unity step size and the calculation (or approximation with lower bounds) of the nadir point guarantee that no Pareto optimal solutions are left undiscovered.

The above calculation procedure is straightforward and can be easily coded either using an external solver with a programming language or in a modeling language. In our case it is coded in GAMS modeling language [12].

3. Computational experiment

The computational experiment comprises the well studied Multiobjective Multidimensional Knapsack Problem (MOMKP) which is the multiobjective version of the well known Multidimensional Knapsack Problem (MKP). The MOMKP is a fundamental MOIP problem. Let us first discuss the formulation of the problems under study.

The general Multi-Objective Integer Programming (MOIP) problem can be formulated as follows:

$$\begin{aligned} &\max\ Cx \\ &s.t.\quad Ax \le b \\ &\qquad x \ge 0 \qquad x \in \mathbf{Z}^n \end{aligned} \tag{2}$$

where C is a $k \times n$ matrix, A is a $m \times n$ matrix and b is a m vector.

The Multi-Objective Multidimensional Knapsack problem (MOMKP) can be formulated as follows:

$$\begin{aligned} &\max\ Cx \\ &s.t.\quad Ax \le b \\ &\qquad x \ge 0 \qquad x \in \{0,1\}^n \end{aligned} \tag{3}$$

where C is a $k \times n$ matrix, A is a $m \times n$ matrix and b is a m vector.

The computational experiment included benchmarks for two objectives and three objectives MKP.

3.1 Two objective MKP

The improved augmented epsilon constraint with jumps (AUGMECON2) presented in this study is compared to the original version of augmented epsilon constraint (AUGMECON) developed in [3], firstly in a test bed of 16 artificial datasets (in total, 32 runs). The structure of these datasets is illustrated in Table 3.

Table 3. The test bed of 16 datasets for two objective MKP

	2 digits for C	3 digits for C
U instances	2kp100	2kp100b
	2kp250	2kp250b
	2kp500	2kp500b
	2kp750	2kp750b
W instances	2kp-W-100	2kp-W-100b
	2kp-W-250	2kp-W-250b
	2kp-W-500	2kp-W-500b
	2kp-W-750	2kp-W-750b

There are 3 parameters in the creation of Table 3 datasets: a) Type of instances b) digits for C matrix and c) number of items, n

a) Type of instances. In the literature only the U type of instances is present ("U" stands for Uncorrelated instances for C and A matrix elements) and especially with 2 digits for C matrix. In this type belong the mostly used data of Zitzler and Laumanns available at [13] and [14] (which appear in the web site http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/) that have been extensively used in the literature. These are called 2kp100, 2kp250, 2kp500 and 2kp750 in our study. They comprise two objectives, two constraints and n=100,250,500,750. These are the standard benchmarks. U type refers to C matrix elements derived from a Uniform distribution, either in U[10,100] (2 digits) or U[100,1000] (3 digits).

In addition to U type instances, we have added according to Shah and Reed in [15] the new type of "W" instances, which stands for Weakly correlated instances for C and A matrix elements. Following Shah and Reed in [15] we have also introduced correlations between C and A matrix elements. For each knapsack i, the technological coefficients a_{ij} are generated randomly between 10 and 100 (e.g. $a_{ij} \in U[10,100]$) and the objective function coefficients $c_{ij} \in U[a_{ij} - 10, a_{ij} + 10]$. This correlation level is a realistic factor because e.g. in project selection settings it is expected that probably higher cost candidate projects may have higher evaluation score in some quality criteria. In general the W type problems are harder to solve than the U type. These datasets with their exact Pareto fronts are available in https://sites.google.com/site/kflorios/augmecon2.

b) Digits for C matrix. It is expected that when larger coefficients for the objective functions are present, the MOIP problem becomes more difficult to be solved exactly, since the pay off table results in an order of magnitude greater range between ideal and nadir point co-ordinates. So, also instances with 3 digits for C matrix elements have been studied which come from the corresponding 2 digit instances by simply adding a small error term $e \in U[0,9]$ and multiplying the scale of the 2 digit instances by a factor of 10. So, the structure of each instance is not altered by a different sampling. We note that only the C coefficients have been refined to 3 digits precision. The A coefficients remain in 2 digits precision, so the constraints and thus feasible region of MOIP remain unchanged. We have only increased precision in the measurement of the objective functions. This makes the 3 digits instances

harder to solve exactly. This is also true even for Dynamic Programming algorithms in single objective single constraint knapsack problems as stated in an example in Papadimitriou and Steiglitz [16], pp.424-425.

c) Number of items, n. Following Zitzler and Laumanns standard datasets, we have restricted our study in the range n=100, 250, 500, 750 items. Also, larger samples have been created with 1000, 1250 and 1500 items, but were not solved because the exact solution of the 750 item benchmarks was already time consuming, and we chose to expand to factors a) and b) previously mentioned and not just expand the size of the instance, n, keeping only U type and 2 digits instances into account. For U type and 2 digits for C coefficients instances, (the Zitzler-Laumanns benchmarks) Lust has made available also the datasets and the Pareto Fronts at his site for the two objectives case https://sites.google.com/site/thibautlust/research/multiobjective-knapsack

3.2 Three objective MKP

The three objective MKP was analysed using only 3 datasets which are illustrated in Table 4.

Table 4. The three datasets for three objective MKP

	2 digits for C
U instances	3kp40
	3kp50
	3kp100

The dataset coding 3kpX means knapsack problems with three objective functions, three constraints and X binary variables. The instances 3kp40 and 3kp50 are taken from Laumanns et al. [17] while 3kp100 is taken from Laumanns et al. [18] and also the Zitzler-Laumanns TestProblemSuite site mentioned in Section 3.1. The instances 3kp40 and 3kp50 have been also solved in Florios et al. [6]. The main interest here is the solution of 3kp100 which is rather challenging. According to Laumanns et al. in [18], 255h (=hours) was the run time for this instance with adaptive epsilon constraint. In this study, the AUGMECON2 method proposed improves greatly in this run time, being roughly an order of magnitude faster.

4. Results and Discussion

The MOMKP model and the augmecon2 method proposed in this paper have been created and solved in GAMS 23.5 environment using CPLEX 12.2 solver. The OS is Windows 7 32-bit and the hardware is a standard core i3 notebook at 2.13 GHz with 4GB RAM for the 4.1 Subsection runs (two objectives MKP) and a standard core i5 notebook at 2.40GHz with 4GB RAM for the 4.2 Subsection runs (three objectives MKP).

4.1 Two objective MKP results

The results from the two objective multidimensional knapsack problems are shown in tables 5-8. The Grid points are actually the second objective function's range increased by one. The Models solved are the number of IP problems solved for each problem, i.e. the model described in (1). |PF*| is the cardinality of the Pareto Front (number of Pareto optimal solutions in the exact Pareto Front)

Table 5. AUGMECON2 and AUGMECON statistics for 2-objective MOMKP (U-Type, 2 digits)

U Type [10,100]	CPU time sec	Grid points	Models solved	|PF*|
AUGMECON2				
2kp100	40	823	144	121
2kp250	932	2534	594	568
2kp500	9601	4176	1654	1416
2kp750	43355	7232	3699	3030
AUGMECON				
2kp100	227	823	823	121
2kp250	2781	2534	2534	568
2kp500	15290	4176	4176	1416
2kp750	55483	7232	7232	3030

Table 6. AUGMECON2 and AUGMECON statistics for 2-objective MOMKP (U-Type, 3 digits)

U Type [100,1000]	CPU time sec	Grid points	Models solved	|PF*|
AUGMECON2				
2kp100b	48.16	8225	136	135
2kp250b	1914	25341	746	732
2kp500b	17132	41774	2396	2332
2kp750b	92913	71966	6143	5868
AUGMECON				
2kp100b	2457	8225	8225	135
2kp250b	30621	25341	25341	732
2kp500b	145686	41774	41774	2332
2kp750b	-	-	-	-

Table 7. AUGMECON2 and AUGMECON statistics for 2-objective MOMKP (W-Type, 2 digits)

W type [10,100]	CPU time sec	Grid points	Models solved	|PF*|
AUGMECON2				
2kp-W-100	309.11	278	111	111
2kp-W-250	4251	666	375	375
2kp-W-500	7514*	1944	968	834
2kp-W-750	12579*	1914	1556	1251
AUGMECON				
2kp-W-100	567.80	278	278	111
2kp-W-250	4593	666	666	375
2kp-W-500	9052*	1944	1944	834
2kp-W-750	13562*	1914	1914	1251

Table 8. AUGMECON2 and AUGMECON statistics for 2-objective MOMKP (W-Type, 3 digits)

W Type [100,1000]	CPU time sec	Grid points	Models solved	|PF*|
AUGMECON2				
2kp-W-100b	816.11	2803	205	197
2kp-W-250b	24032	7267	1678	967
2kp-W-500b	46569*	19527	3256	2980
2kp-W-750b	128454*	19619	5529	4960
AUGMECON				
2kp-W-100b	6765	2803	2803	197
2kp-W-250b	59831	7267	7267	967
2kp-W-500b	130022*	19527	19527	2980
2kp-W-750b	-	-	-	-

Where an asterisk (*) is noted, this means that four threads of CPLEX have been used for the IP sub-problems (otherwise only one thread of CPLEX has been used). Dashes "-" in the cells mean that the specific runs did not terminate after 48 hours.

Tables 5-8 report on the CPU time, Grid points, Models Solved and |PF*| of the corresponding datasets. By looking at Table 5 results, we see a significant decrease of CPU time and Models Solved in favour of AUGMECON2 over original AUGMECON. This advantage is attributed to the jumps introduced (see section 2.1) which skip redundant gridpoints. This is even more apparent in the results of Table 6, where the digits of C matrix are three, and the gridpoints are ten times more, due to larger integer coefficients. The models solved for AUGMECON2 in Table 6 are driven by |PF*|, being a desirable feature of the algorithm, while the models solved for the original AUGMECON are driven by the gridpoints' number, which is very high, due to the large coefficients present. The example of Figure 2 is characteristic. It depicts the computational time and the "models solved" parameters for the four 2kp problems with large coefficients (data from Table 6).

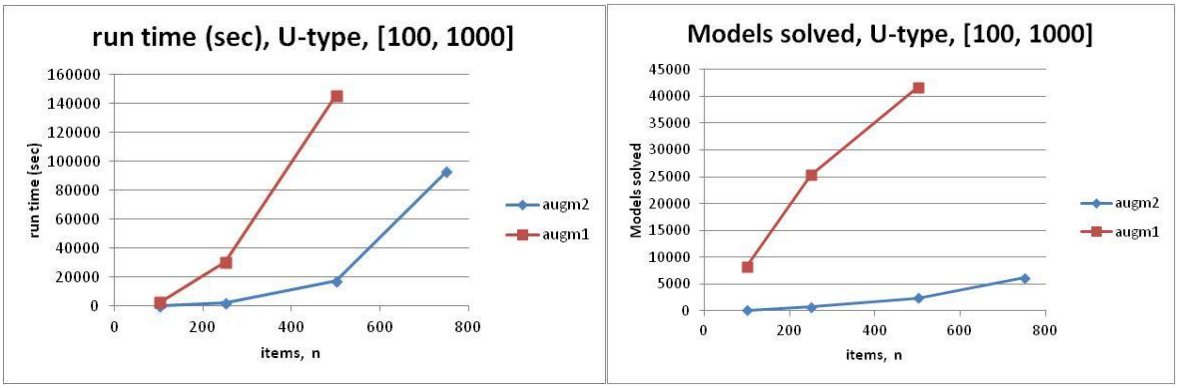

Figure 2. Computational time and "models solved" for the 2kp problems of U-type with objective function coefficients in [100-1000]

This is a key finding of this study, which justifies the improved version of AUGMECON2. Also, the differences are overwhelming in the CPU time in Table 6 between the two versions. AUGMECON2 is

found to be an order of magnitude faster than the original AUGMECON, and also AUGMECON2 solves the hardest 2kp750b dataset, whilst the original AUGMECON cannot solve it in 2 days limit.

By looking at Table 7 results, we evaluate the newly introduced W type instances. Firstly, the size of the Pareto front in these instances seems to be less than the size of the Pareto front in corresponding U type instances, with the same number of digits for C elements. Also, while for smaller number of items, W type instances seem to be harder than U type instances, the 4 thread use of CPLEX makes even the larger 500 and 750 item datasets affordable. Comparing AUGMECON2 to the original AUGMECON, we see that in these datasets, AUGMECON2 is marginally faster again. Table 8 instances are probably the hardest to solve since both complicating factors are present, W type and 3 digits for C elements. Again, we note that the number of Models solved for AUGMECON2 is driven by the |PF*| number, whilst for original AUGMECON the models solved are driven by the gridpoints which eventually result in a prohibitive number, due to the large coefficients present. AUGMECON2 again solves a dataset, namely 2kp-W-750b that the original AUGMECON cannot solve in 2 days limit. In Table 8 results, AUGMECON2 is 2-3 times faster than the original AUGMECON.

4.2 Three objective MKP results

Table 9 presents the results for the exact solution of U type instances with 2 digits for C in three objectives and three constraints. From Section 4.1, one concludes that AUGMECON2 dominates AUGMECON in all benchmarks with two objectives and two constraints. This is by far confirmed in the three objective cases. Additionally, in this section we compare AUGMECON2 with method ADECON of Laumanns et al. [17, 18] and MCBB [6, 19]. For large instances ADECON is clearly better than MCBB. Nevertheless, for small to medium instances (up to 30-40 binary variables), MCBB remains the fastest method, but it failed to provide a solution for the 100 binary variables (stopped after 48h). It should be noticed that there is a dominated objective vector among the 6501 Laumanns et al. report on their website. Our results conform with those of Lust at his web site. We find exactly 6500 non dominated vectors for 3kp100. The nadir vectors of these problems, which are necessary for the implementation of AUGMECON2, were given as the exact Pareto front of these benchmarks is available in the literature. As it was mentioned, the runs of AUGMECON2 and AUGMECON were made by the authors in a core i5 2.4GHz and the results of the previous methods (computational times of ADECON and MCBB) are adjusted according to the benchmarks of Dongarra [20] and the use of LINPACK utility for standardization of system performance.

Table 9. AUGMECON2 statistics and comparison with AUGMECON, Adecon, and MCBB for 3-objective MOMKPs (U-Type)

U Type [10,100]	CPU time	Grid points per obj. function	Models solved	|PF*|
AUGMECON2				
3kp40	37 min	540	7802	389
3kp50	159 min	846	24903	1048
3kp100	33 h	1236	103049	6500
AUGMECON				
3kp40	15 h	540	242386	389
3kp50	41 h	846	489746	1048
3kp100	dnt*	1236	dnt	dnt
ADECON [18]				
3kp40	29 min	-	26846	389
3kp50	209 min	-	128695	1048
3kp100	120 h	-	644689	6501
MCBB [6]				
3kp40	6.5 min	-	-	389
3kp50	164 min	-	-	1048
3kp100	dnt	-	dnt	dnt

*dnt = did not terminate within 48 hours which means problem too big for the method

5. Concluding remarks

The aim of this paper is to propose and evaluate an enhancement of the original augmented ε-constraint method which is especially suitable to cope with MOIP problems. Specifically, the new version AUGMECON2 proved to be very efficient in providing the exact Pareto set in MOIP problems compared to the previous version and some other methods in the literature. The basic innovation in AUGMECON2 in relation to the original version is the introduction of the bypass coefficient. The bypass coefficient exploits the slack/surplus information of the innermost constrained objective function and skips an often considerable number of grid points before it proceeds to the next call to the solver. This is done with no cost whatsoever to the accuracy of the algorithm as only redundant iterations resulting in the same Pareto optimal solution are skipped. So, being a win-win feature, the bypass coefficient is proven a valuable characteristic of AUGMECON2 over AUGMECON as shown in the computational experiment with a well known MOIP problem. In the computational experiment we used known benchmarks found in the literature as well as some new benchmarks that are published for first time. The problem in which the performance of the proposed method is evaluated is a well studied MOIP problem, namely the Multi-Objective Multidimensional Knapsack Problem (MOMKP).

Regarding the MOMKP, the proposed method solved successfully difficult bi-objective instances of the MOMKP, both uncorrelated and weakly correlated. It greatly improves the original version AUGMECON and this is more apparent where larger integer coefficients for the objective functions are present. The key finding is that the "models solved" metric for AUGMECON2 is linear in the size

of the Pareto front in all bi-objective MOMKP instances, while for the original AUGMECON method the "models solved" metric is proportional to the number of grid points which is actually the range of the payoff table for the constrained objective. Especially in the case of 3 digit objective function coefficients (i.e. in the range [100, 1000]) the economies of AUGMECON2 over AUGMECON are impressive. As it is obvious, the 3 digit objective function coefficient problems are harder to solve than the regular 2 digit coefficients problems. In addition, it is also noticed that the newly introduced W type instances are harder to solve than the regular U type instances, although the cardinality of their Pareto front is noticeably smaller. The three objective problems are much more challenging than bi-objective problems. For three objectives and three constraints, only up to 100 items are solved, in high computational time. We tested AUGMECON2 in benchmark problems found in the literature and we compared it also with two other exact methods, namely, the adaptive ε-constraint (ADECON) and MCBB. The results were favourable for AUGMECON2 especially in the larger instances.

Future research includes some methodological issues. The methodological issues have to do with the further exploitation of the information provided in every iteration of the ε-constraint method (to create bypass rules not only for the innermost loop) and the safe calculation of lower bounds for the nadir values. Parallelization of the process is also within our future plans as well as testing of the method in other MOIP problems like e.g. Multi-Objective Set Covering problems, Multi-Objective Shortest Path problems and Multi-Objective Spanning Tree problems.

References

[1] R.E. Steuer, Multiple Criteria Optimization.Theory, Computation and Application, Krieger, Malabar FL, 1986.

[2] C.L. Hwang, A. Masud, Multiple Objective Decision Making. Methods and Applications: A state of the art survey, Lecture Notes in Economics and Mathematical Systems Vol. 164, Springer-Verlag, Berlin, 1979.

[3] G. Mavrotas, Effective implementation of the ε-constraint method in multi-objective mathematical programming problems, Applied Mathematics and Computation 213 (2009) 455-465.

[4] C.A. Coello Coello, D.A. Van Veldhuizen, G.A. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, Boston, MA, 2002.

[5] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley and Sons, Chichester, 2001.

[6] K. Florios, G. Mavrotas, D. Diakoulaki, Solving multi-objective, multi-constraint knapsack problems using mathematical programming and evolutionary algorithms, European Journal of Operational Research 203 (2010) 14-21.

[7] H. Isermann, R.E. Steuer, Computational experience concerning payoff tables and minimum criterion values over the efficient set, European Journal of Operational Research 33 (1987) 91-97.

[8] M.I. Dessouky, M. Ghiassi, W.J. Davis, Estimates of the mínimum nondominated criterion values in multiple criteria decisión making, Engineering Costs and Production Economics 10 (1986) 95-104.

[9] B. Metev, V. Vassilev, A method for nadir point estimation in MOLP problems, Cybernetics and Information Technologies 3 (2003) 15-24.

[10] M.J. Alves, J.P. Costa, An exact method for computing the nadir values in multiple objective linear programming, European Journal of Operational Research 198 (2009) 637-646.

[11] J.M. Jorge, An algorithm for optimizing a linear function over an integer efficient set, European Journal of Operational Research 195 (2009) 98-103.

[12] A. Brooke, D. Kendrick, A. Meeraus, R. Raman, GAMS. A user's guide, GAMS development corporation, Washington, 1998.

[13] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, IEEE Transactions on Evolutionary Computation 3 (1999) 257-271.

[14] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, in: K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, T. Fogarty (Eds.), Evolutionary Methods for Design, Optimization, and Control, Barcelona, CIMNE , Spain, 2002, pp. 19-26.

[15] R. Shah, P. Reed, Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems, European Journal of Operational Research 211 (2011) 466-479.

[16] C.H. Papadimitriou, K. Steiglitz, Combinatorial optimization: algorithms and complexity, Dover, New York, 1998.
[17] M. Laumanns, L. Thiele, E. Zitzler, An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method, European Journal of Operational Research 169 (2006) 932-942.
[18] M. Laumanns, L. Thiele, E. Zitzler, 2005. An adaptive scheme to generate the Pareto front based on the epsilon-constraint method, in: J. Branke, K. Deb, K. Miettinen, R. Steuer (Eds.), Practical Approaches to Multi-Objective Optimization, Dagstuhl Seminar Proceedings, Vol. 04461, URN: urn:nbn:de:0030-drops-2465, URL: http://drops.dagstuhl.de/opus/volltexte/2005/246.
[19] G. Mavrotas, D. Diakoulaki, Multi-criteria branch & bound: A vector maximization algorithm for Mixed 0-1 Multiple Objective Linear Programming, Applied Mathematics and Computation 171 (2005) 53-71.
[20] J.J. Dongarra, Performance of Various Computers Using Standard Linear Equations Software, Linpack Benchmark Report, University of Tennessee Computer Science Technical Report, CS-89-85, 2008.