Operational Research (2022) 22:1291–1332
https://doi.org/10.1007/s12351-020-00574-6

ORIGINAL PAPER

A robust augmented ε-constraint method (AUGMECON-R) for finding exact solutions of multi-objective linear programming problems

Alexandros Nikas[1] · **Angelos Fountoulakis**[1] · **Aikaterini Forouli**[1] · **Haris Doukas**[1]

Received: 15 April 2020 / Revised: 15 April 2020 / Accepted: 14 May 2020 / Published online: 24 May 2020

Abstract

Systems can be unstructured, uncertain and complex, and their optimisation often requires operational research techniques. In this study, we introduce AUGMECON-R, a robust variant of the augmented ε-constraint algorithm, for solving multi-objective linear programming problems, by drawing from the weaknesses of AUGMECON 2, one of the most widely used improvements of the ε-constraint method. These weaknesses can be summarised in the ineffective handling of the true nadir points of the objective functions and, most notably, in the significant amount of time required to apply it as more objective functions are added to a problem. We subsequently apply AUGMECON-R in comparison with its predecessor, in both a set of reference problems from the literature and a series of significantly more complex problems of four to six objective functions. Our findings suggest that the proposed method greatly outperforms its predecessor, by solving significantly less models in emphatically less time and allowing easy and timely solution of hard or practically impossible, in terms of time and processing requirements, problems of numerous objective functions. AUGMECON-R, furthermore, solves the limitation of unknown nadir points, by using very low or zero-value lower bounds without surging the time and resources required.

Keywords Augmecon · ε-constraint · Multi-objective programming · Optimisation · Pareto

✉ Alexandros Nikas
anikas@epu.ntua.gr

[1] Decision Support Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens, Iroon Politechniou 9, 157 80 Athens, Greece

1 Introduction

Despite rapid technological advancements in software and hardware performance, many problems featuring numerous evaluation criteria or objective functions (Wiedemann 1978), multiple constraints of different nature and hundreds to thousands of decision variables remain challenging to solve (Carrizosa et al. 2019). Several methods have been developed to solve multi-objective linear programming problems, each of which features strengths and weaknesses (Sylva and Crema 2007). Among these, the ε-constraint method, along with its variants, has been used in many systems and applications (e.g. Liu and Papageorgiou 2013; Paul et al. 2017; Zhou et al. 2018; Jenkins et al. 2019), reported as a powerful way to solve multi-objective linear programming problems and preferred over competing techniques (Kadziński et al. 2017a, b; Jabbarzadeh et al. 2019).

In this study, we focus on one of the most widely used improvements of ε-constraint, the AUGMECON method (Mavrotas 2009) as subsequently improved in AUGMECON 2 (Mavrotas and Florios 2013). By looking at its main novelties, its core weaknesses are identified and discussed in detail, serving as a motivation for developing a new model that effectively overcomes them. These weaknesses, although dependent on various characteristics and processes of the method, can be summarised in the ineffective handling of the true nadir points of the objective functions of a problem and, most notably, in the significant amount of time required to apply it as more objective functions are added to a problem, which can even make a problem practically insolvable. Drawing on these, we introduce AUGMECON-R, a powerful and robust improvement that addresses these weaknesses, and apply it in comparison with its predecessor, in both a set of reference problems from the literature and a series of significantly more complex problems of four to six objective functions.

The rest of the paper is organised as follows. Section 2 carries out a brief overview of the ε-constraint, AUGMECON and AUGMECON 2 methods, by highlighting their characteristics of significance to introducing AUGMECON-R in Sect. 3. Section 4 performs a comparative analysis between AUGMECON-R and its predecessor (AUGMECON 2). Finally, Sect. 5 draws conclusions and outlines prospects for future work.

2 A brief overview of the augmented ε-constraint method

According to Hwang et al. (1980), Multiple-Objective Mathematical Programming (MOMP) solving algorithms can be organised in three groups: a priori methods, in which the decision makers have the capacity to express their preferences or objective function weights prior to solving the problem; interactive methods, which feature an ongoing dialogue between analysts and decision makers, eventually leading to preferences converging with solutions; and a posteriori methods, in which the problem is solved and the effective Pareto solutions are

found, allowing the decision makers to select among these based on their preferences. Given the infrequency of early knowledge and quantification capacity of the decision makers' preference model, which is prerequisite to a priori methods, and the difficulty in the decision makers having complete overview of (an approximation of) the Pareto front, associated with interactive methods, this paper orients on a posteriori methods to solving a MOMP problem of the form:

$$\max \left\{f_1(x), f_2(x), \ldots, f_p(x)\right\}$$
$$s.t. : x \in S$$

where x is the vector of decision variables, $f_1(x), f_2(x), \ldots, f_p(x)$ are the p objective functions, and S is the space of efficient solutions.

Among these methods, the ε-constraint algorithm aims at optimising one objective function, while considering all other objective functions as constraints. The model, widely applied for multi-objective linear programming problems (Mavrotas et al. 2011; Sakar and Koksalan 2013), is thus transformed to:

$$\max \left\{f_1(x)\right\}$$
$$s.t. :$$
$$f_2(x) \geq e_2$$
$$f_3(x) \geq e_3$$
$$\ldots$$
$$f_p(x) \geq e_p$$
$$x \in S$$

By changing the right-hand side of the constrained objective functions (e_i), efficient solutions are obtained. The problem is solved on a step-by-step basis on an $N_2 \times N_3 \times \ldots \times N_p$ grid, where N_i is the integer range of the objective function f_i.

One of the method's main advantages is that the number of efficient solutions can be controlled, by appropriately adjusting the number of grid points, on which each optimisation is solved, along the range of each objective function. However, this range must be calculated; it cannot be secured that solutions are not weak but effective; and solving any problem with more than two objective functions is very time-consuming.

These weaknesses motivated the development of augmented ε-constraint or AUGMECON (Mavrotas 2009), which transforms the problem into the following:

$$\max \left\{ f_1(x) + eps \times \left(s_2 + s_3 + \cdots + s_p \right) \right\}, eps \in \left(10^{-6}, 10^{-3} \right)$$

$$s.t. :$$

$$f_2(x) - s_2 = e_2$$

$$f_3(x) - s_3 = e_3$$

$$\cdots$$

$$f_p(x) - s_p = e_p$$

$$x \in S$$

In essence, AUGMECON introduces the following modifications to the original ε-constraint method, to ensure that only effective Pareto solutions are obtained: (i) all constraints corresponding to the $p - 1$ objective functions become strict inequalities; and (ii) slack (or surplus) variables are introduced both to the primary objective function and to the constrained ones.

Another significant novelty of AUGMECON is that it exploits cases where the problem is infeasible, leading to an early exit from the nested loop of the step increase function: the algorithm initially sets lower bounds to the constrained objective functions, which gradually become stricter; if the problem becomes infeasible, i.e. the model cannot be solved for the given constraint of an objective function, after a specific grid point increase, there is no point in strengthening the constraints and the algorithm exits from the innermost loop and continues to the next grid point of said objective function. This way AUGMECON contributes to faster model solution, especially when the problem features more than two objective functions.

AUGMECON has been employed in various applications and systems, including supply chain management (Torabi et al. 2013; Bootaki et al. 2014, 2016; Canales-Bustos et al. 2017; Musavi and Bozorgi-Amiri 2017; Ravat et al. 2017; Vieira et al. 2017; Sazvar et al. 2018; Ehrenstein et al. 2019; Oiu et al. 2019; Shekarjan et al. 2019; Xin et al. 2019), energy planning (Hombach and Walther 2015; Tartibu et al. 2015; Arancibia et al. 2016; Cambero and Sowlati 2016; Cambero et al. 2016; Mohammadkhani et al. 2018; Rabbani et al. 2018; Sedighizadeh et al. 2018; Razm et al. 2019), waste management (Mavrotas et al. 2013; Mavrotas et al. 2015a, b; Inghels et al. 2016), portfolio analysis (Xidonas et al. 2010, 2011; Khalili-Damghani et al. 2012), transportation (Resat and Turkay 2015; Babakeik et al. 2018) and others (Khalili-Damghani and Amiri 2012; Aras and Yurdakul 2016; Yu et al. 2018; Behmanesh and Zandieh 2019; Zhang et al. 2019; Xiong et al. 2019); and has been combined with or compared against evolutionary algorithms (Khalili-Damghani et al. 2013; Dabiri et al. 2017; Wang et al. 2018; Mohammadi et al. 2019).

Mavrotas and Florios (2013) further extended this algorithm in AUGMECON 2, by introducing a bypass coefficient as well as a type of lexicographic optimisation to all objective functions, the order of which was insignificant in AUGMECON:

$$\max \left\{ f_1(x) + eps \times \left(s_2/r_2 + 10^{-1} s_3/r_3 + \cdots + 10^{-(p-2)} s_p/r_p \right) \right\}$$

By means of the bypass coefficient, AUGMECON 2 makes use of the information provided by the slack/surplus variables of the constrained objective functions to

avoid unnecessary iterations and accelerate solution. The jumps made in the innermost loop to help accelerate grid scanning allow for decreasing the step of the process and therefore increasing the grid points; by doing so, the exact Pareto set can be identified. But, in order to do so, (a) the objective function coefficients must be integer and (b) the nadir points of the Pareto set must be known.

To deal with the first limitation, non-integer coefficients can be multiplied by the appropriate power of 10, as necessary, which can however significantly expand the grid and increase the grid points, leading to very large solution times. Regarding the second limitation, adding steps to accurately calculate the nadir points of the Pareto set can also increase the algorithm's complexity, so the AUGMECON 2 algorithm only uses an underestimation (overestimation), i.e. a lower (upper) bound, in cases of maximisation (minimisation) objectives.

Despite its weaknesses, which are analysed in detail below, AUGMECON 2 has significantly better performance over AUGMECON, and this is why it has also been applied in a wide range of problem domains since its introduction, including supply chain management (Gavranis and Kozanidis 2017; Bal and Satoglu 2018; Attia et al. 2019; Habibi et al. 2019; Resat and Unsal 2019; Roshan et al. 2019; Saedinia et al. 2019; Vafaeenezhad et al. 2019; Mohammed and Duffuaa 2020), project selection (Mavrotas et al. 2015a, b; Schaeffer and Cruz-Reyes 2016), and network optimisation and planning (Florios and Mavrotas 2014; Oke and Siddiqui 2015; Mousazadeh et al. 2018; Rahimi et al. 2019), as well as in policy-related problems, such as energy and climate action (Forouli et al. 2019a, b; Van de Ven et al. 2019; Doukas and Nikas 2020).

3 Augmecon-r

3.1 Motivation

Although AUGMECON 2 constitutes a significant upgrade to AUGMECON and a powerful algorithm for solving multi-objective integer programming (MOIP) problems and finding the exact Pareto set of a problem, it features certain weaknesses, the need to overcome which has motivated the development of AUGMECON-R.

First, the solution time for large-scale problems of more than twoobjective functions is still high, since jumps only occur in the innermost loop and not across the grid and for all nested loops, which represent the constrained objective functions: a problem of m objective functions of average range n would have an AUGMECON 2 complexity of $O\left(n^{m-1}\right)$, which is relatively large for programs running in environments like GAMS. For example, a 6kpY problem (a knapsack problem of 6 objective functions, 6 constraints and Y decision variables), with an average integer range of 1000 for each objective function, would feature a complexity of $O\left(10^{15}\right)$, or slightly less given the iterations avoided due to the bypass coefficient of the innermost loop. The more objective functions a problem has, the more time-consuming AUGMECON 2 becomes for solving said problem.

Second, AUGMECON 2 requires that objective function coefficients be integer. If this is not the case, non-integer coefficients are multiplied by the appropriate power

of 10, thereby also increasing the complexity accordingly: a problem of m objective functions of average range n and an average number of decimals k would have an AUGMECON 2 complexity of $O\left(n^{m-1} \times 10^{k\times(m-1)}\right)$.

Third, implementing AUGMECON 2 requires a priori knowledge of the nadir points of the objective functions. Nadir point calculation algorithms are usually complex, hard to program and could require writing chunks of code larger than those of AUGMECON 2 itself; are generally capable of solving problems of up to three objective functions; and their running time is comparable to the one required by AUGMECON 2 (Alves and Costa 2009). This is why AUGMECON 2 opts for underestimation of nadir points, i.e. the use of lower bounds of the objective functions, thereby only slightly increasing computation time. This process of approximating the nadir points, in AUGMECON 2, takes place in the problem's payoff table where the lowest values, which in theory are equal to or greater than the nadir points, are multiplied by an *arbitrary* coefficient (e.g. 90%), resulting in what is *hopefully* an underestimation of the actual nadir points. Academically speaking, one heuristic approach around this would be the calculation of all payoff tables, considering all possible orders of the constrained objective functions; this would expectedly give a closer approximation to the actual nadir points, allowing tightening the *arbitrary* coefficient, e.g. to 95%, *hopefully* ensuring that the nadir point would be included in the new, smaller grid. However, this approach would simply improve computation time, without avoiding either the arbitrary or the hopeful nature of the approximation process.

Fourth, the correlation between the order of constrained objective functions across loops and computation time is a weakness by itself: AUGMECON 2 features a bypass coefficient only for the innermost loop of the process, resulting in getting rid of only those unnecessary grid point checks that can be avoided within the innermost loop. In order to maximise the number of unnecessary iterations avoided as much as possible, after calculating the payoff table, the algorithm should be in a position to switch order of constrained objective functions accordingly, so that the objective function of the largest range could be placed in the innermost loop.

These four limitations associated with AUGMECON 2 constitute the motivation of developing a new algorithm that can significantly improve computation time and efficiency, as well as allow for easily solving problems that have so far been hard or practically impossible to solve.

3.2 An improved search algorithm

Reading through (Mavrotas and Florios 2013) and the performance recorded for AUGMECON 2, there appears to be a large deviation between the number of models solves and the solutions included in the Pareto front. For example, in the case of the 3kp100 problem—i.e. of a knapsack problem of three objective functions, three constraints and a hundred decision variables—there are 103,049 models solved, which is approximately sixteen times the number of the solutions included in the Pareto Front (6,500). However, given that this is a MOIP problem and AUGMECON 2 calculates the exact Pareto set by using a unity step to

explore all possible integer values of the objective functions across the grid, one would expect that the models solved would be equal or at least close to the number of Pareto front solutions, which is not the case.

This large number of unnecessary optimisations computed can be attributed to the use of only one bypass coefficient in the innermost loop and, in addition, to the large number of infeasibilities that could have otherwise been to some extent foreseen and avoided.

In this direction, AUGMECON-R introduces a novelty that is largely based on the existing notion of the bypass coefficient, by incorporating to the model as many bypass coefficients as objective functions, which would be of the form:

$$\begin{aligned} b_2 &= int\left(s_2/step_2\right) \\ b_3 &= int\left(s_3/step_3\right) \\ &\cdots \\ b_p &= int\left(s_p/step_p\right) \end{aligned}$$

where $int()$ is the function that returns the integer part of a real number, and s_i is the slack/surplus variable for an objective function i, and $step_i$ is the discretisation step for this objective function:

$$step_i = r_i/q_i$$

where r_i is the range of the objective function i, and q_i the number of equal intervals that the range is divided to formulate the grid, so that the latter comprise $q_i + 1$ grid points.

This way, instead of having one bypass coefficient acting at the innermost loop like AUGMECON 2, AUGMECON-R features an active bypass coefficient in each one of the outer loops as well. In every iteration, bypass coefficients $b_i = int\left(s_i/step_i\right)$ are calculated. When $s_i > step_i$, in the next iteration for b_i' corresponding to $e_i' = e_i + step_i$ the optimisation will again lead to the same solution, with $s_i' = s_i - step_i$, making the iteration unnecessary. The b_i bypass coefficient indicates how many iterations should be bypassed, provided that these iterations concern the i^{th} objective function and the right-hand sides of all other constrained objective functions remain constant. The new process introduced in the proposed algorithm can be shown with a simple example. Assume that we have a four-objective problem with the following payoff table as shown in Table 1 (all objective functions to be maximised):

From the payoff table, we have $r_2 = r_3 = r_4 = 10$, which are divided into ten equal intervals, with a unity step of $step_2 = step_3 = step_4 = 1$. AUGMECON-R includes the following process:

Table 1 Payoff table of example problem

	f_1	f_2	f_3	f_4
max $f_1(x)$	105	102	77	50
max $f_2(x)$	95	112	80	53
max $f_3(x)$	100	108	87	46
max $f_4(x)$	100	110	80	56

Table 2 Grid points of the example problem

Objective function	Counter	Grid points										
		0	1	2	3	4	5	6	7	8	9	10
$f_2(x)$	k	102	**[103]**	104	105	106	107	108	109	110	111	112
$f_3(x)$	j	77	78	79	**[80]**	81	82	83	84	85	86	87
$f_4(x)$	i	46	47	**[48]**	49	50	51	52	53	54	55	56

$$
\begin{aligned}
&For\ i = 0 - 10\\
&\quad e_4 = 46 + i\\
&\quad For\ j = 0 - 10\\
&\qquad e_3 = 77 + j\\
&\qquad For\ k = 0 - 10\\
&\qquad\quad e_2 = 102 + k\\
&\qquad\quad Solve(P)\\
&\qquad Next\ k\\
&\quad Next\ j\\
&Next\ i
\end{aligned}
$$

The objective function $f_2(x)$ is represented in the innermost loop (k counter). Assume that we currently are at the 2nd iteration of the innermost loop ($k = 1$), the 4th iteration of the middle loop ($j = 3$)and the 3rd iteration of the outermost loop ($i = 2$), with $e_2 = 103$, $e_3 = 80$, and $e_4 = 48$, as displayed in brackets and bold in Table 2.

After the optimisation, we obtain $s_2 = 4$, $s_3 = 3$, and $s_4 = 4$, meaning that $f_2 = 103 + 4 = 107$, $f_3 = 80 + 3 = 83$, and $f_4 = 48 + 4 = 52$ (and, for the sake of completeness, $f_1 = 97$). Hence, $b_2 = 4$, $b_3 = 3$, and $b_4 = 4$. While AUGMECON 2 wouldconsider unnecessary only the four next iterations of the innermost loop, AUGMECON-R acknowledges that any combination of $k \in [1, 5]$, $j \in [3, 6]$,

$i \in [2, 6]$ would return the same solution. In this problem, AUGMECON-R would avoid 19 unnecessary iterations that AUGMECON 2 would not.

Assuming we have the capacity to store the values of the bypass coefficients b_i in optimisation h and defining as pure any optimisation that leads to a solution different from the one resulting from a unity decrease of any of the parameters for the right-hand side for a specific iteration drawn from the grid points of the objective functions, e_i, then AUGMECON-R can avoid:

$$\sum_{h \in D} \{b_{3,h}\}, \quad if\ p = 3$$

$$\sum_{h \in D} \{b_{3,h} + b_{4,h} * (b_{3,h} + 1)\}, \quad if\ p = 4$$

$$\sum_{h \in D} \{b_{3,h} + b_{4,h} * (b_{3,h} + 1) + b_{5,h} * (b_{4,h} + 1)(b_{3,h} + 1)\}, \quad if\ p = 5$$

$$\sum_{h \in D} \{b_{3,h} + b_{4,h}(b_{3,h} + 1) + b_{5,h}(b_{4,h} + 1)(b_{3,h} + 1) + \ldots + b_{p,h}(b_{p-1,h} + 1)(b_{p-2,h} + 1) \ldots (b_{3,h} + 1)\}, \quad if\ p \geq 6$$

iterations compared to AUGMECON 2, where: h is a pure optimisation and D is the sum of all pure optimisations.

In order to achieve this for any problem of p objective functions, as suggested above, the AUGMECON-R algorithm requires that a $(p-1)$-dimensional array be introduced to store integer flag values, $flag[(N_2 + 1) \times (N_3 + 1) \times \cdots \times (N_p + 1)]$, where N_i is the integer range of the objective function f_i. The array is initialised with zero values and, prior to any optimisation, the algorithm examines if the corresponding value of the array is zero or not; if it is zero, the optimisation is performed, otherwise the algorithm jumps in the innermost loop as many steps as the array value indicates.

By introducing the flag array and the notion of pure optimisations, AUGMECON-R can at the same time avoid any unnecessary optimisations due to infeasibilities: if, for any value of $e_2^*, e_3^*, \ldots, e_p^*$ of the right-hand side of the constrained objective functions, there lies an infeasibility, then for an increase of any of $e_2, e_3, \ldots, e_p$ with all others equal to or greater than $e_2^*, e_3^*, \ldots, e_p^*$ an infeasibility will also be reached. Therefore, for $\delta_i \in N$, any $\{e_i^* + \delta_i\}$ combination on the right-hand side of the constrained objective functions will return an infeasibility; while AUGMECON 2 would only avoid infeasibilities for $\{e_2^* + \delta_2, e_3^*, \ldots, e_p^*\}, \delta_2 > 0$, AUGMECON-R avoids all infeasibilities for $\{e_2^* + \delta_2, e_3^* + \delta_3, \ldots, e_p^* + \delta_p\}$.

The proposed algorithm can, therefore, avoid all iterations, for which all right-hand sides of the constrained objective functions are equal to or greater than the e_i^* values that led to an infeasibility.

The flow chart of AUGMECON-R is shown in Fig. 1.

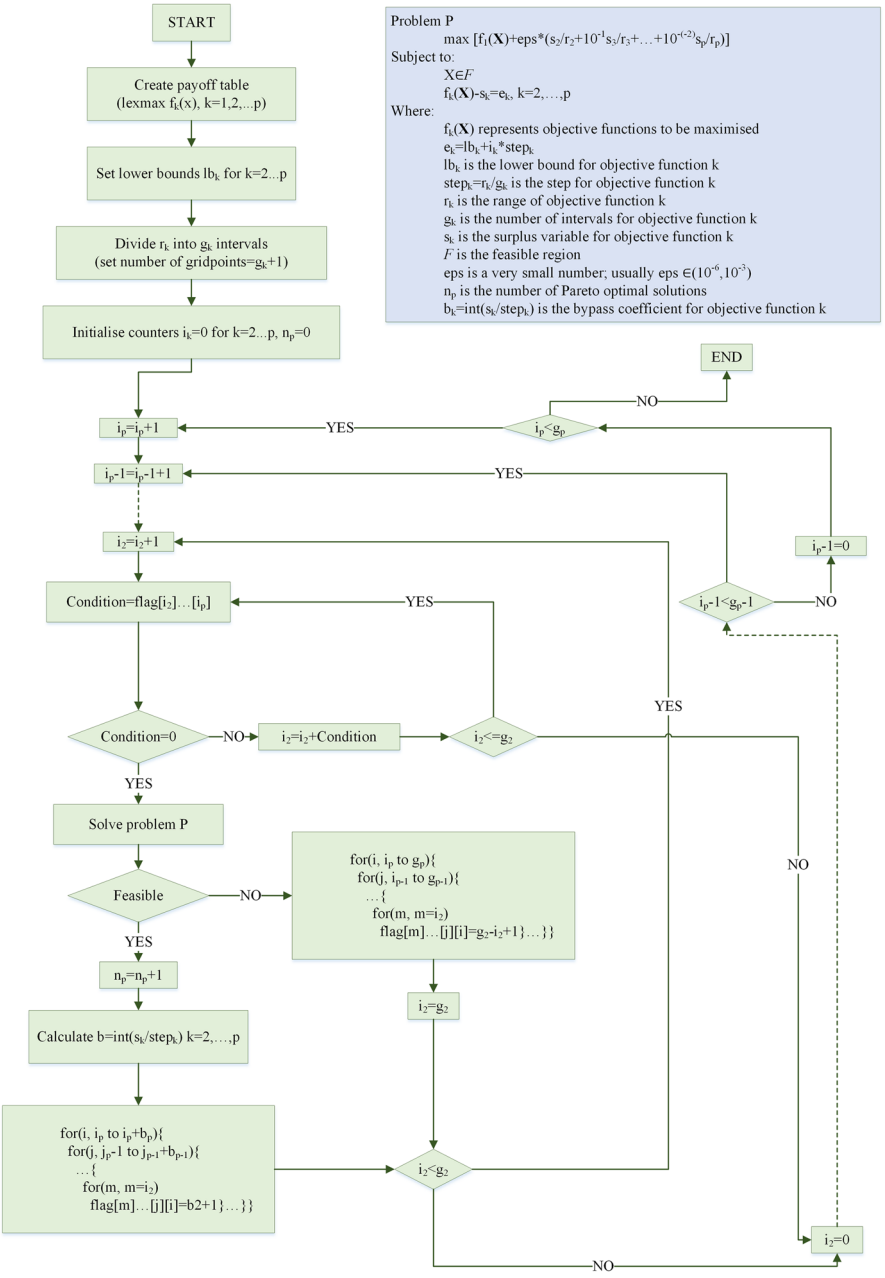

Fig. 1 Flowchart of the AUGMECON-R algorithm

3.3 Source code

The code for AUGMECON-R customised for a representative model of a 4kp40 problem, freely available on GitHub[1], has largely been based on the AUGMECON 2 source code and is presented in Appendix 1.

4 Comparative analysis and discussion

In this section, AUGMECON-R is employed for numerous problems and its performance is compared against the performance of AUGMECON 2. Initially, the benchmark problems presented by Mavrotas and Florios (2013) are solved, acting as a reference, followed by a series of random, more challenging in terms of objective functions and density problems; for the latter, AUGMECON 2 was also used by the authors, to provide for a comparative analysis.

It must be noted that all problems presented in the section have been solved in GAMS version 23.5, using CPLEX 12.2, a 64-bit Windows 10 operating system, a 2.7 GHz i5-6400 processor and an 8GB RAM memory.

4.1 Reference benchmark problems

Here, given that AUGMECON-R was designed as an upgrade to AUGMECON 2, we use as reference the 3kpY problems Mavrotas and Florios (2013) used to compare the performance of and establish AUGMECON 2 against AUGMECON; these include a 3kp100, a 3kp50 and a 3kp40 problem, i.e. selected knapsack problems of three objective functions, three constraints, and 100, 50 and 40 decision variables respectively. The 2kpY problems used in the same study were disregarded since, based on the proposed model outlined in Sect. 3.2, AUGMECON-R is identical to AUGMECON2 when dealing with only two objective functions.

It should also be noted that, in their study, Mavrotas and Florios (2013) do not use their originally proposed AUGMECON 2 algorithm, but a programming modification of it that arbitrarily avoids certain optimisations at the initial stages, both in the innermost loop and in the outer loop. This is noteworthy as, although the use of this modification does not significantly change the order of the resulting difference (cf. the performance reported in Mavrotas and Florios 2013), here the performance of AUGMECON-R is compared against the original AUGMECON 2 algorithm, and not against the ad hoc modified one. Table 3 summarises the performance between the two algorithms, in terms of the CPU time needed, the grid points per objective function, the total models solved, the infeasibilities found, the number of models solved multiple times (‘duplicate solutions’), the dominated solutions and the solutions found in the Pareto front.

[1] https://github.com/KatforEpu/Augmecon-R

Table 3 Performance comparison between AUGMECON 2 (AUGM 2) and AUGMECON-R (AUGM-R) for the 3kpY problems

	3kp100		3kp50		3kp40	
	AUGM 2	AUGM-R	AUGM 2	AUGM-R	AUGM 2	AUGM-R
CPU Time	23 h	268 min	113 min	695 s	42 min	220 s
Grid points per objective function	1236	1236	846	846	540	540
Models solved	103,652	11,727	25,245	1951	11,098	746
Infeasibilities	1093	137	564	78	420	34
Duplicate solutions	96,020	5071	23,630	823	10,287	321
Dominated solutions	39	19	3	2	2	2
Solutions in the Pareto front	6500	6500	1048	1048	389	389

Table 4 Comparison ratios of performance of AUGMECON 2 over AUGMECON-R for the 3kpY problems

Problem	CPU time ratio	Models solved ratio	Infeasibilities ratio	Duplicate solutions ratio
3kp100	5.15	8.84	7.98	18.93
3kp50	9.75	12.94	7.23	28.71
3kp40	11.45	14.88	12.35	32.05

Our findings suggest that, for the same number of solutions in the Pareto front, AUGMECON-R is multiple times faster and solves significantly less models, leading to significantly fewer infeasibilities and duplicate solutions (Table 4). To make up for potential randomness in CPU times due to different levels of CPU core availability, the CPU times presented are average times after a series of model runs, so that comparison can be considered unbiased and representative. This is also why the number of models solved is highlighted as a comparison metric, indicating similar ratios. It should be noted that the differences in CPU time ratios and models solved can be attributed to the time needed by AUGMECON-R to perform the bypass condition checks. Furthermore, the differences of ratios among the three problems can be attributed to the different density of the problems, i.e. the ratio of the number of solutions included in the Pareto front over the number of models solved: the denser the problem, the smaller the time difference between the two algorithms, as fewer iterations are avoided in the loops outside the innermost loop.

To highlight the enhanced performance of AUGMECON-R over AUGMECON 2, the arbitrary selection of the lower bounds loosens, to maximise the probability of including the actual nadir points in the analysis and ensure that no solution is missed. So, instead of multiplying the nadir values of the payoff tables by 95%, as was the case in the problems above, we reiterate our analysis of these three problems, by multiplying the nadir values by 5%, leading to an emphatically larger grid,

Table 5 Performance comparison between AUGMECON 2 (AUGM 2) and AUGMECON-R (AUGM-R) for the 3kpY problems with lower bounds

	3kp100*		3kp50*		3kp40*	
	AUGM 2	AUGM-R	AUGM 2	AUGM-R	AUGM 2	AUGM-R
CPU time	62 h	274 min	230 min	737 s	130 min	234 s
Grid points per objective function	3940	3940	1880	1880	1560	1560
Models solved	417,809	11,768	61,442	1953	39,648	748
Infeasibilities	1093	137	564	78	420	34
Duplicate solutions	410,138	5090	59,827	825	38,836	322
Dominated solutions	78	41	3	2	3	3
Solutions in the Pareto front	6500	6500	1048	1048	389	389

in order to evaluate how this impacts the performance of the two algorithms in comparison (Table 5).

Although the difference of the two algorithms is now more evident for the case of significantly lower bounds, by looking at Tables 3 and 5, it is worth pointing out that this problem modification led to a CPU time increase of 2.24%, 6.00% and 6.40% for AUGMECON-R, compared to a CPU time increase of 170.00%, 103.50% and 209.52% for AUGMECON 2, for 3kp100, 3kp50 and 3kp40 respectively. Similar findings can be observed for all other relevant metrics; for example, the additional models solved by AUGMECON-R are negligible in all three problems (41, 2 and 2), the same cannot be said for AUGMECON 2 (314,157, 36,197 and 28,550 respectively).

4.2 Complex benchmark problems

Here, we implement both algorithms to evaluate AUGMECON-R in problems of more than three objective functions. Before doing so, however, we distinguish between uncorrelated and weakly correlated problems (Martello and Monaci 2020; Shah and Reed 2011): uncorrelated problems assume no correlation between elements of the objective function coefficient matrix and those of the constraint coefficient matrix, while weakly correlated problems assume a weak correlation between these elements. This weak correlation makes their solution significantly more difficult, as the solver requires more time resources, and given the time requirements for AUGMECON 2 to solve such problems, only uncorrelated problems are assumed in this study.

We define the following problems:

- A 4kp40 problem, with 155, 119 and 121 being the true nadir points of the three constrained objective functions and 123, 127 and 140 being their ranges respectively.
- A 4kp40* problem, which is identical with the 4kp40 problem but without a priori knowledge of nadir points, hence with the consideration of significantly

Table 6 Performance comparison between AUGMECON 2 and AUGMECON-R for the 4kp40 problem, with the true nadir points (4kp40) and with lower bounds (4kp40*)

	4kp40		4kp40*	
	AUGMECON 2	AUGMECON-R	AUGMECON 2	AUGMECON-R
CPU time	1214 min	56 min	85 h	59min
Models solved	290,443	14,735	1,431,195	10,846
Infeasibilities	14,735	359	35,363	359
Duplicate solutions	272,530	7324	1,392,653	7315
Dominated solutions	6	0	7	0
Solutions in the Pareto front	3172	3172	3172	3172

lower bounds: 15, 11 and 13 being the lower bounds and 263, 235 and 248 the range respectively.

- A 4kp50 binary problem, with objective function coefficients resulting from a uniform distribution $U[0, 1]$ and constraint coefficients from a uniform distribution $U[50, 70]$, with 718, 735 and 713 being the true nadir points, and 51, 35 and 44 being the ranges respectively.
- A 4kp50* binary problem, which is identical with the 4kp50 binary problem but after extending the range of the objective functions by assigning new lower bounds at 70, 69 and 57.
- A 5kp40 problem, with objective function coefficients resulting from a uniform distribution $U[50, 40]$ and constraint coefficients from a uniform distribution $U[2, 10]$, with 29, 32, 27 and 27 being the true nadir points of the four constrained objective functions, and 21, 21, 27 and 25 being the ranges respectively.
- A 5kp40* problem, which is identical with the 5kp40 problem but after extending all ranges to 45, to make sure we include the now unknown true nadir points, with 5, 8, 9 and 7 being the new lower bounds.
- A 6kp50 binary problem, with objective function coefficients resulting from a uniform distribution $U[0, 1]$ and constraint coefficients from a uniform distribution $U[0, 5]$, with 38, 37, 31, 27 and 30 being the true nadir points of the five constrained objective functions, and 21, 24, 26, 30 and 22 being the ranges respectively.
- A 6kp50* problem, which is identical with the 6kp50 binary problem but after extending all ranges to 50, to make sure we include the now unknown true nadir points, with 9, 11, 7, 7 and 2 being the new lower bounds.

The matrices of the objective function and constraint coefficients are provided in Appendix 2, for all of the above pairs of problems, i.e. for 4kp40 – 4kp40*, 4kp50 – 4kp50*, 5kp40 – 5kp40* and 6kp50 – 6kp50*.

Table 6 summarises the performance differences between AUGMECON 2 and AUGMECON-R, for the problems 4kp40 and 4kp40*, while Fig. 2 visualises the Pareto front of the problem.

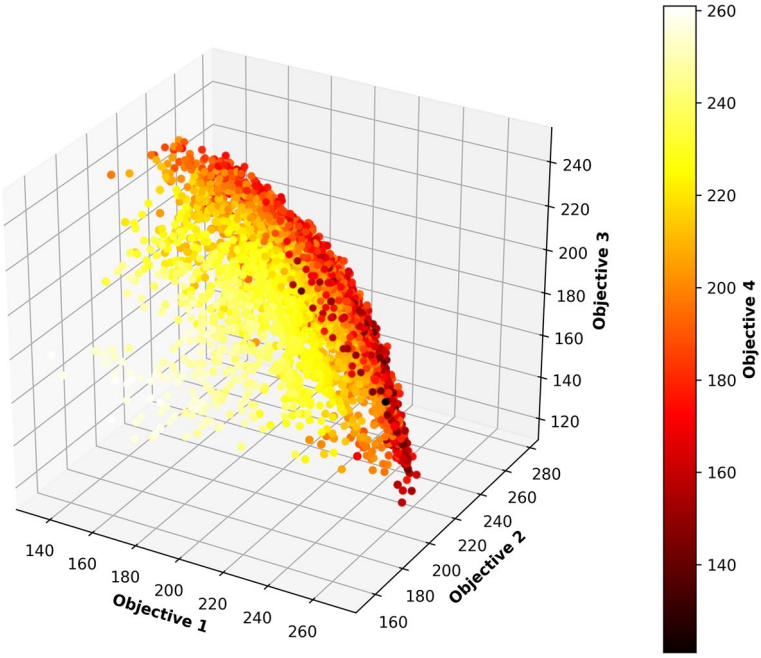

Fig. 2 The Pareto front of the 4kp40 problem

Table 7 Performance comparison between AUGMECON 2 and AUGMECON-R for the 4kp50 problem, with the true nadir points (4kp50) and with lower bounds (4kp50*)

	4kp50		4kp50*	
	AUGMECON 2	AUGMECON-R	AUGMECON 2	AUGMECON-R
CPU Time	1021 s	31 s	161 h	939 s
Models solved	6296	176	> 4,000,000	161
Infeasibilities	1211	28	–	28
Duplicate solutions	5039	102	–	87
Dominated solutions	0	0	–	0
Solutions in the Pareto front	46	46	46	46

It is evident, from Table 4, that AUGMECON-R is almost 21 times faster than its predecessor, with the latter solving almost 26 times more models, in the problem where the actual nadir points are known a priori; this ratio difference is, as discussed above, due to the number of checks made by AUGMECON-R in its *flag* array. When considering the case of the true nadir points being unknown and thus extending the grid to secure that the actual nadir points are included in the analysis and that no solution is missed, AUGMECON-R outperforms AUGMECON 2 by solving about 131 times less models, more than 85 times faster. One odd finding is

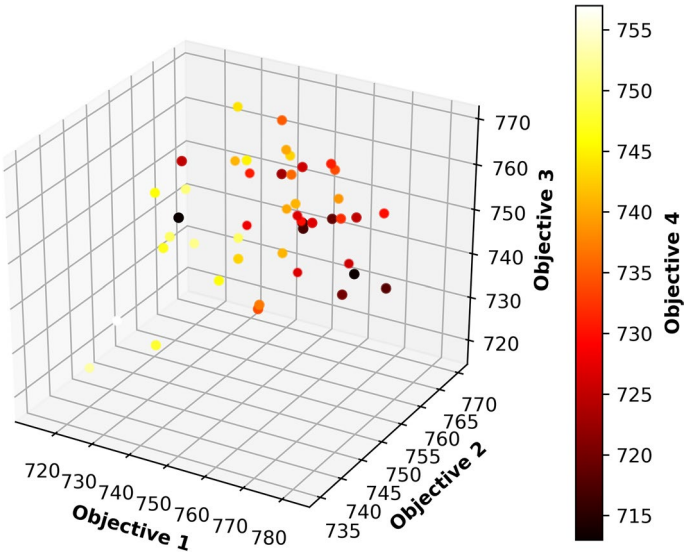

Fig. 3 The Pareto front of the 4kp50 problem

Table 8 Performance comparison between AUGMECON 2 and AUGMECON-R for the 5kp40 problem, with the true nadir points (5kp40) and with lower bounds (5kp40*)

	5kp40		5kp40*	
	AUGMECON 2	AUGMECON-R	AUGMECON 2	AUGMECON-R
CPU time	12,035 s	175 s	27 h	194 s
Models solved	52,030	618	458,760	622
Infeasibilities	9351	114	47,521	114
Duplicate solutions	42,553	378	411,113	382
Dominated solutions	0	0	0	1
Solutions in the Pareto front	126	126	126	126

that AUGMECON-R now solves even less models than before; given how small the lower bounds are, the surplus variables are significantly larger, and this circumstantially leads to less models. This, however, does not bear any negative impacts on the accuracy of the algorithm, as it stumbles upon equally as many infeasibilities.

Similarly, Table 7 summarises the performance differences between AUGMECON 2 and AUGMECON-R, for the binary problems 4kp50 and 4kp50*, while Fig. 3 visualises the Pareto front of the problem.

Again, AUGMECON-R solves the 4kp50 problem almost 32 times faster, having solved about 35 times less models. But what is strikingly interesting is that, when considering the 4kp50* problem, AUGMECON 2 required 161 hours and solved more than four million models. These findings clearly indicate that

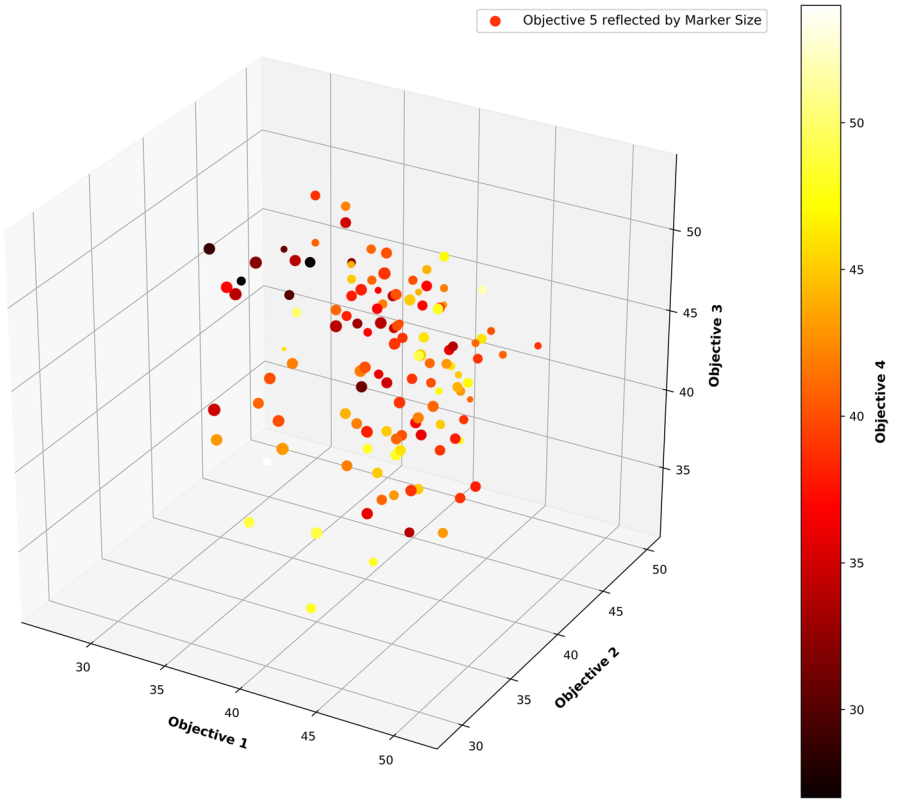

Fig. 4 The Pareto front of the 5kp40 problem

AUGMECON-R has the capacity to timely solve time-wise non-viable, complex problems, ensuring accuracy and assuring no solution is missed.

Moving onto a five-objective problem, Table 8 summarises the performance differences between AUGMECON 2 and AUGMECON-R, for the problems 5kp40 and 5kp40*, while Fig. 4 visualises the Pareto front of the problem.

AUGMECON-R solved the 5kp40 problem almost 68 times faster, having solved about 83 times less models, while in the case of lack of a priori knowledge lower bounds considering the 4kp50* problem, these differences surge to 505 and 737 times respectively. In both cases, however, it is evident that AUGMECON-R outperforms AUGMECON 2 even more than in the previous two sets of problems of four objective functions; as previously discussed, the larger the number of objective functions is, the larger this outperformance is. This can be highlighted in the final problem of six objective functions, as follows in Table 9 and Fig. 5.

As with the 4kp50 binary problem, the 6kp50 binary problem solved by both AUGMECON 2 and AUGMECON-R emphasises the performance difference between the two methods. Given the significantly higher complexity that a sixth objective function adds to the problem, the CPU time and models solved ratios

Table 9 Performance comparison between AUGMECON 2 and AUGMECON-R for the 6kp50 problem, with the true nadir points (6kp50) and with lower bounds (6kp50*)

	6kp50		6kp50*	
	AUGMECON 2	AUGMECON-R	AUGMECON 2	AUGMECON-R
CPU time	52 h	1207 s	–	4145 s
Models solved	1,104,406	6269	–	6242
Infeasibilities	193,612	863	–	863
Duplicate solutions	909,949	4563	-	4536
Dominated solutions	2	0	–	0
Solutions in the Pareto front	843	843	843	843

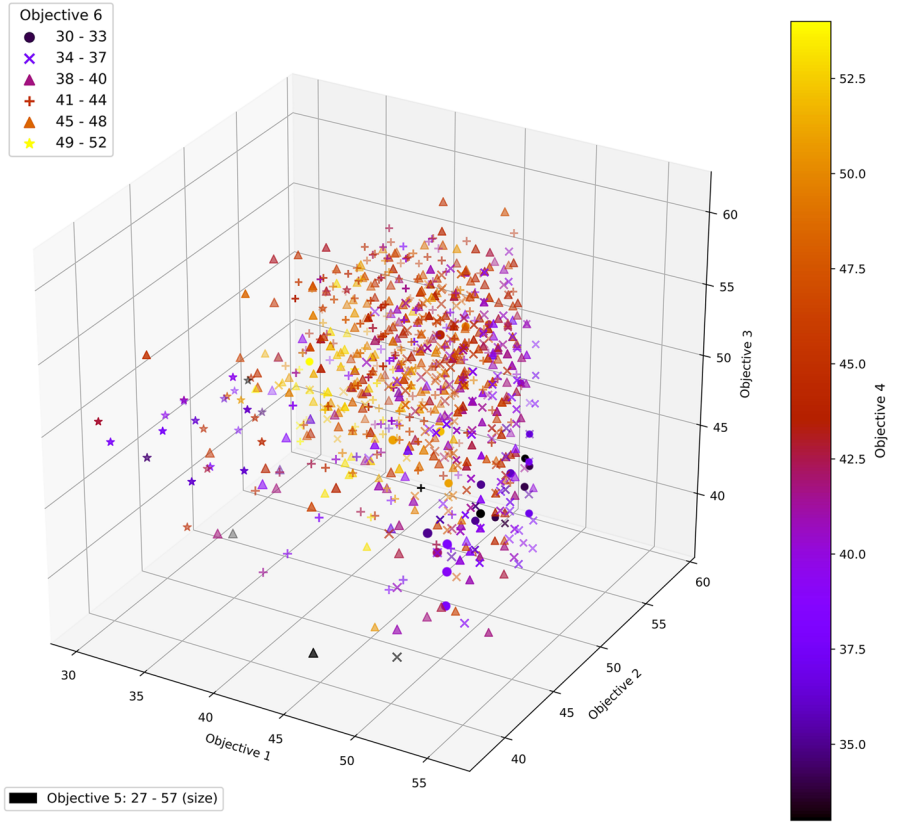

Fig. 5 The Pareto front of the 6kp50 problem

are even higher in this case, with AUGMECON 2 solving 175 more models 155 times slower. By extending the grid in order to ensure that the a priori unknown true nadir points are included in the analysis, AUGMECON 2 cannot solve the

problem in a reasonable amount of time—it took the algorithm 47 hours to cross the grid once.

5 Conclusions

In this study, an improved version of the augmented ε-constraint method, AUGMECON-R, is introduced, allowing robust and timely optimisation of complex systems. Drawing from the weaknesses associated with its predecessor (AUGMECON 2), the concept and mathematical model of the proposed method is presented in detail and its code provided in Appendix 1, before implementing both methods in comparison. The problems solved in Sects. 4.1, 4.2 suggest that the proposed method, AUGMECON-R, greatly outperforms its predecessor, AUGMECON 2, by solving significantly less models in emphatically less time and allowing us to easily and timely solve hard or even impossible, in terms of time and processing requirements, problems of multiple objective functions. AUGMECON-R, furthermore, solves the problem of unknown nadir points, by using very low or zero-value lower bounds without increasing the time requirements.

As with ε-constraint (e.g. Ehrgott and Ryan 2002; Laumanns et al. 2006), there exist in the literature a few other attempts to identify weaknesses associated with and improve accordingly the AUGMECON 2 algorithm (e.g. Domínguez-Ríos et al. 2019), which however tend to perform a posteriori and numerous checks that potentially enhance complexity and time requirements, such as (Zhang and Reimann 2014), which additionally is developed in Visual Studio Express instead of the usual operational research problem solving implementation platform, GAMS.

One limitation of the proposed method lies in the introduction of a *flag* array, the size of which is directly linked to the range of the objective functions and therefore can lead to occupy a large memory space that could be unavailable. To overcome this, AUGMECON-R could in the future be developed in an object-oriented language like C++, instead of GAMS, which allows for dynamic memory allocation. This would enable using virtual memory, avoiding the *flag* array initialisation with zero values and releasing memory space whenever a counter moves across the grid.

Given that even the slightest uncertainty in the data can render system optimality meaningless from a practical point of view (Bertsimas and Sim 2004) and given recent advancements on robust linear optimisation (Bertsimas and Brown 2009), other prospects for future research include the all-in-one integration of AUGMECON-R with uncertainty and robustness analysis methods (Van de Ven et al. 2019; Mastorakis and Siskos 2016; Ben-Tal et al. Ben-Tal et al. 2010; Kadzinski et al. 2017a, b; Witting et al. 2013), thereby avoiding the use of numerous methods, code scripts, or even implementation platforms.

Acknowledgements The most important part of this research is based on the H2020 European Commission Project "PARIS REINFORCE" under grant agreement No. 820846. The sole responsibility for the content of this paper lies with the authors. The paper does not necessarily reflect the opinion of the European Commission.

Funding European Commission Horizon 2020 Framework, 'PARIS REINFORCE' Research and Innovation Project, Grant Agreement No. 820846.

Compliance with ethical standards

Conflict of interest Not applicable.

Availability of data and material Data available in the Appendix, on request and on Github: https://github.com/KatforEpu/Augmecon-R.

Code availability Open source, code available on Github: https://github.com/KatforEpu/Augmecon-R

Appendix 1: Source code of AUGMECON-R

```
Set
   I 'constraints'         / i1* i4 /
   J 'decision variables'  / j1*j40 /
   K 'objective functions' / k1* k4 /
   ;

Parameter
   dir(k) 'direction of the objective functions 1 for max and -1 for
min' / k1 1, k2 1, k3 1, k4 1  /
   b(I)   'RHS of the constraints' /   i1 1570, i2 1210 , i3 1355, i4
1035/;

Table c(J,K) 'matrix of objective function coefficients C'
           k1        k2        k3        k4
j1         7         22        17        5
j2         13        10        11        25
j3         16        20        5         8
j4         19        20        11        18
j5         24        20        3         20
j6         24        3         7         10
j7         23        24        4         7
j8         6         7         19        20
j9         5         24        8         17
j10        20        16        8         11
j11        10        24        3         10
j12        7         14        7         15
j13        23        20        9         2
j14        3         8         15        20
j15        7         3         16        23
j16        20        19        19        18
j17        9         10        10        10
j18        13        4         12        5
j19        20        2         12        4
j20        18        17        13        11
j21        17        10        12        23
j22        6         10        7         24
j23        7         15        19        8
j24        10        7         11        15
j25        11        12        24        12
j26        5         7         22        8
j27        22        10        5         3
j28        16        17        21        21
j29        16        7         13        16
j30        3         10        14        5
j31        8         23        24        11
j32        3         11        4         19
j33        20        10        5         2
j34        18        15        7         9
j35        10        4         5         19
j36        22        9         8         21
j37        6         19        13        8
j38        20        10        10        3
j39        12        24        17        6
j40        11        24        16        21
```

```
  ;
Table a(J,I) 'matrix of technological coefficients A'
            i1          i2          i3          i4
j1          78          59          53          76
j2          94          67          75          51
j3          97          88         117          88
j4         116         107         101         102
j5          50          65          77          90
j6          62          77          88         114
j7          66          93          52         107
j8         110          89          64          94
j9          63         107         118          57
j10         59         110          87          71
j11        118          95          66          58
j12        104          77         101         114
j13        117         111         116         106
j14        120          97         105          94
j15         65         100          65         109
j16        102          95          97          73
j17        100          69          84          81
j18         97          99          55          77
j19         61          66          99          53
j20        102         113         103          85
j21         71          89         115          71
j22         86          73          91          99
j23         53          85          98          56
j24        110          88          64          84
j25         58          84         113         101
j26         87          58          60          50
j27         69          76          83          69
j28         69          79         111          83
j29         71          96          81         113
j30         83          75          64          94
j31         85         112         110          84
j32         88          81          80          75
j33        109          63          61          71
j34        115         103          56          80
j35        106         112          69         105
j36         95          68          75          76
j37         98          71          71          83
j38         87          52          52          80
j39        102          94         109          54
j40         56         107          63         101
   ;
Variable
   Z(K) 'objective function variables'
   X(J) 'decision variables';

Binary Variable X;
```

```
Equations
   objfun(K)     objective functions
   con(I)     constraints
;

objfun(K).. sum(J,c(J,K)*X(J)) =e= Z(K);
con(I).. sum(J, a(J,I)*X(J)) =l= b(I);

Model example / all /;

$STitle eps-constraint method

Set k1(k) the first element of k, km1(k) all but the first elements
of k;
k1(k)$(ord(k)=1) = yes; km1(k)=yes; km1(k1) = no;
Set kk(k)      active objective function in constraint allobj  ;

Parameter
   rhs(k)      right hand side of the constrained obj functions in
eps-constraint
   maxobj(k)   maximum value from the payoff table
   minobj(k)   minimum value from the payoff table
   numk(k) ordinal value of k starting with 1
   range(k) maxobj-minobj   ;

Scalar
iter   total number of iterations
infeas total number of infeasibilities
elapsed_time elapsed time for payoff and e-sonstraint
start start time
finish finish time      ;

Variables
   a_objval    auxiliary variable for the objective function
   obj         auxiliary variable during the construction of the
payoff table ;
Positive Variables
   sl(k)       slack or surplus variables for the eps-constraints   ;
Equations
   con_obj(k) constrained objective functions
   augm_obj    augmented objective function to avoid weakly efficient
solutions
   allobj      all the objective functions in one expression;

con_obj(km1)..    z(km1) - dir(km1)*sl(km1) =e= rhs(km1);

* We optimize the first objective function and put the others as
constraints
* the second term is for avoiding weakly efficient points

* objfun=max z1 + 0.001*(s1/r1+0.1 s2/r2+ 0.01*s3/r3+...)
augm_obj..
  sum(k1,dir(k1)*z(k1))+1.0e-3*sum(km1,power(10,-(numk(km1)-
1))*sl(km1)/(maxobj(km1)-minobj(km1))) =e= a_objval;
```

```
allobj..  sum(kk, dir(kk)*z(kk)) =e= obj;

Model mod_payoff    / example, allobj / ;
Model mod_epsmethod / example, con_obj, augm_obj / ;

Parameter
  payoff(k,k)  payoff tables entries;
Alias(k,kp);

option optcr=0.0;
option limrow=0, limcol=0, solprint=off, solveLink =
%solveLink.LoadLibrary% ;
$offlisting;
$offsymxref;
$offsymlist;
$offuelxref;
$offuellist;
*,solveLink = %solveLink.LoadLibrary%
*file cplexopt /cplex.opt/;
*put cplexopt;
*put 'threads 4'/;
*put 'parallelmode 1'/;
*putclose cplexopt;
*mod_epsmethod.optfile=1;
*option optca=0.;
*mod_payoff.optfile=1;
*mod_epsmethod.optfile=1;

* Generate payoff table applying lexicographic optimization
loop(kp,
  kk(kp)=yes;
  repeat
    solve mod_payoff using mip maximizing obj;
    payoff(kp,kk) = z.l(kk);
    z.fx(kk) = z.l(kk);
    kk(k++1) = kk(k);
  until kk(kp); kk(kp) = no;
* release the fixed values of the objective functions for the new
iteration
  z.up(k) = inf; z.lo(k) =-inf;
);
if (mod_payoff.modelstat<>1 and mod_payoff.modelstat<>8, abort 'no
optimal solution for mod_payoff');

File fx  / 4kp40_uncorrelated_nadir.txt /;

PUT fx ' PAYOFF TABLE'/   ;
loop (kp,
        loop(k, put payoff(kp,k):12:2);
        put /;
     );
put fx /;

*display payoff;
*minobj(k)=smin(kp,payoff(kp,k));
**$ontext
```

```
*Ideally minobj(k) could be set to zero value minobj(k)=0;
minobj(k)=floor(0.088*smin(kp,payoff(kp,k)));
maxobj(k)=smax(kp,payoff(kp,k));
range(k)=(maxobj(k)-minobj(k));
*$ontext
*$set fname h.%screxT.dat%

*gridpoints=max integer of km1

$if not set gridpoints_1  $set gridpoints_1 1000
$if not set gridpoints_2  $set gridpoints_2 1000
$if not set gridpoints_3  $set gridpoints_3 1000
*Generally speaking gridpoints are set to a very large value
Set g grid points /g0*g%gridpoints%/
    grid(k,g) 'grid '
    q    /q0*q%gridpoints_1%/
    r     /r0*r%gridpoints_2%/
    s    /s0*s%gridpoints_3%/

   ;

Parameter
    gridrhs(k,g)
    maxg(k) maximum point in grid for objective
    posg(k) grid position of objective
    firstOffMax, lastZero, current1, current2, current3,  synthiki,
b2, b3, b4, terminal1, terminal2, terminal3, control1, control2,
range1, range2, range3 some counters
*    numk(k) ordinal value of k starting with 1
    numg(g) 'ordinal value of g starting with 0  '
    step(k) step of grid points in objective functions
    jump(k) jumps in the grid points' traversing only for the first
objective function
    numq(q)  ordinal value of q starting with zero
    numr(r)  ordinal value of r starting with zero
    nums(s)  ordinal value of s starting with zero
    flag(q,r,s) memory matrix

;

 lastZero=1; loop(km1, numk(km1)=lastZero; lastZero=lastZero+1);
numg(g) = ord(g)-1;
 numq(q) = ord(q)-1;
 numr(r) = ord(r)-1;
 nums(s) = ord(s)-1;
 range1=sum(k$(ord(k)=2),range(k));
 range2=sum(k$(ord(k)=3),range(k));
 range3=sum(k$(ord(k)=4),range(k));

loop(q$(numq(q)<range1+1),
       loop(r$(numr(r)<range2+1),
            loop(s$(nums(s)<range3+1), flag(q,r,s)=0);));
```

```
grid(km1,g) = yes;
maxg(k)=range(k);
step(km1)   = 1 ;
gridrhs(grid(km1,g))$(dir(km1)=-1) = maxobj(km1) -
numg(g)/maxg(km1)*(maxobj(km1)- minobj(km1));
gridrhs(grid(km1,g))$(dir(km1)=1) = minobj(km1) + numg(g)*step(km1);
*display gridrhs;

*PUT fx ' Grid points'/   ;
*loop (g,
*       loop(km1, put gridrhs(km1,g):12:2);
*        put /;
*     );
put fx /;
put fx 'Efficient solutions'/;

* Walk the grid points and take shortcuts if the model becomes
infeasible
posg(km1) = 0;

iter=0;
infeas=0;
terminal1=0;
terminal2=0;
terminal3=0;
terminal1=sum(km1$(numk(km1)=1),maxg(km1));
terminal2=sum(km1$(numk(km1)=2),maxg(km1));
terminal3=sum(km1$(numk(km1)=3),maxg(km1));
synthiki=0;
control1=0;
start=jnow;

repeat
  rhs(km1) = sum(grid(km1,g)$(numg(g)=posg(km1)), gridrhs(km1,g));
  current1=0;
  current2=0;
  current3=0;
  current1=sum(km1$(numk(km1)=1),posg(km1))  ;
  current2=sum(km1$(numk(km1)=2),posg(km1))  ;
  current3=sum(km1$(numk(km1)=3),posg(km1))  ;
 loop(q$(numq(q)=current1),
      loop(r$(numr(r)=current2),
         loop(s$(nums(s)=current3), synthiki=flag(q,r,s));););
  if(synthiki=0,  solve mod_epsmethod maximizing a_objval using mip);
  iter=iter+1;
  if (synthiki=0 and mod_epsmethod.modelstat<>1 and
mod_epsmethod.modelstat<>8,
    infeas=infeas+1;
    put fx iter:5:0, '  infeasible'/;
    lastZero = 0; loop(km1$(posg(km1)>0 and lastZero=0),
lastZero=numk(km1));
    posg(km1)$(numk(km1)<=lastZero) = maxg(km1);
       loop(s$(nums(s)>=current3 and nums(s)<=terminal3),
           loop(r$(numr(r)>=current2 and numr(r)<=terminal2),
```

```
                loop(q$(numq(q)=current1), flag(q,r,s)=terminal1-
current1+1);););

  else if(synthiki=0 ,
    put fx iter:5:0;
    loop(k, put fx z.l(k):12:2);
    put fx ' *** ';
    loop(km1, put fx sl.l(km1):12:2, put fx posg(km1):6:0);
    put fx ' *** ';
   loop(km1$(numk(km1)=1),b2=floor(sl.l(km1)/step(km1)));
   loop(km1$(numk(km1)=2),b3=floor(sl.l(km1)/step(km1)));
   loop(km1$(numk(km1)=3),b4=floor(sl.l(km1)/step(km1)));
        loop(s$(nums(s)>=current3 and nums(s)<=current3+b4),
              loop(r$(numr(r)>=current2 and numr(r)<=current2+b3),
                   loop(q$(numq(q)=current1) ,
flag(q,r,s)=b2+1);););


     jump(km1)=1;

* calculate only for the first constrained objective function
jump(km1)
    put fx ' * ';
*    loop(km1$(numk(km1)=1), jump(km1)=1+floor(sl.L(km1)/step(km1)));
    jump(km1)$(numk(km1)=1)=1+floor(sl.L(km1)/step(km1));
    loop(km1, put fx jump(km1):5:0) ;
    loop(km1$(jump(km1)> 1),  put '   jump')
    put /;
    );
   );

 jump(km1)$(numk(km1)>1)=1 ;
* Proceed forward in the grid
  control1=0;
  firstOffMax = 0;
  loop(km1$(posg(km1)<maxg(km1) and firstOffMax=0 and numk(km1)=1 and
synthiki>0),control2=posg(km1)+synthiki;
posg(km1)=min(posg(km1)+synthiki,maxg(km1)); firstOffMax=numk(km1) );
  loop(km1$(posg(km1)=maxg(km1) and numk(km1)=1 and synthiki>0 and
firstOffMax>0 and control2>maxg(km1)), control1=1);
  loop(km1$(posg(km1)<maxg(km1) and firstOffMax=0 and numk(km1)=1 and
synthiki=0), posg(km1)=min((posg(km1)+jump(km1)),maxg(km1));
firstOffMax=numk(km1));
  loop(km1$(posg(km1)<maxg(km1) and firstOffMax=0 and numk(km1)>1),
posg(km1)=min((posg(km1)+jump(km1)),maxg(km1));
firstOffMax=numk(km1));


   loop(km1$(posg(km1)<maxg(km1) and control1>0 and numk(km1)>1 ),
posg(km1)=min((posg(km1)+jump(km1)),maxg(km1));
firstOffMax=numk(km1); control1=0);

                        posg(km1)$(numk(km1)<firstOffMax) = 0  ;

until sum(km1$(posg(km1)=maxg(km1)),1)= card(km1) and firstOffMax=0;
```

```
finish=jnow;
elapsed_time=(finish-start)*86400;

put /;
put 'Infeasibilities = ', infeas:5:0 /;
put 'Elapsed time: ',elapsed_time:10:2, ' seconds' / ;
*$offtext
putclose fx;
**$offtext
```

Appendix 2: Datasets used for the complex problems

Dataset of the 4kp40 problem

Table c(J,K) 'matrix of objective function coefficients C'

	k2	k4	k3	k1
j1	7	22	17	5
j2	13	10	11	25
j3	16	20	5	8
j4	19	20	11	18
j5	24	20	3	20
j6	24	3	7	10
j7	23	24	4	7
j8	6	7	19	20
j9	5	24	8	17
j10	20	16	8	11
j11	10	24	3	10
j12	7	14	7	15
j13	23	20	9	2
j14	3	8	15	20
j15	7	3	16	23
j16	20	19	19	18
j17	9	10	10	10
j18	13	4	12	5
j19	20	2	12	4
j20	18	17	13	11
j21	17	10	12	23
j22	6	10	7	24
j23	7	15	19	8
j24	10	7	11	15
j25	11	12	24	12
j26	5	7	22	8
j27	22	10	5	3
j28	16	17	21	21
j29	16	7	13	16
j30	3	10	14	5
j31	8	23	24	11
j32	3	11	4	19
j33	20	10	5	2
j34	18	15	7	9
j35	10	4	5	19
j36	22	9	8	21

```
j37         6          19         13         8
j38         20         10         10         3
j39         12         24         17         6
j40         11         24         16         21
Table a(J,I) 'matrix of constraint coefficients A'
            i1         i2         i3         i4
j1          78         59         53         76
j2          94         67         75         51
j3          97         88         117        88
j4          116        107        101        102
j5          50         65         77         90
j6          62         77         88         114
j7          66         93         52         107
j8          110        89         64         94
j9          63         107        118        57
j10         59         110        87         71
j11         118        95         66         58
j12         104        77         101        114
j13         117        111        116        106
j14         120        97         105        94
j15         65         100        65         109
j16         102        95         97         73
j17         100        69         84         81
j18         97         99         55         77
j19         61         66         99         53
j20         102        113        103        85
j21         71         89         115        71
j22         86         73         91         99
j23         53         85         98         56
j24         110        88         64         84
j25         58         84         113        101
j26         87         58         60         50
j27         69         76         83         69
j28         69         79         111        83
j29         71         96         81         113
j30         83         75         64         94
j31         85         112        110        84
j32         88         81         80         75
j33         109        63         61         71
```

j34	115	103	56	80
j35	106	112	69	105
j36	95	68	75	76
j37	98	71	71	83
j38	87	52	52	80
j39	102	94	109	54
j40	56	107	63	101

B.2 Dataset of the 4kp50 binary problem

Table c(J,K) 'matrix of objective function coefficients C'

	k1	k3	k2	k4
j1	68	65	65	66
j2	66	50	63	69
j3	59	53	57	62
j4	55	68	69	68
j5	57	51	58	60
j6	67	56	63	70
j7	55	62	53	56
j8	54	64	53	59
j9	57	67	59	65
j10	64	50	62	66
j11	68	59	58	54
j12	62	70	69	50
j13	53	60	67	65
j14	70	62	60	58
j15	52	64	51	63
j16	55	64	53	53
j17	64	56	61	53
j18	52	61	57	57
j19	65	63	70	57
j20	57	69	63	67
j21	61	56	57	61
j22	54	68	61	59
j23	50	64	52	68
j24	57	67	64	52
j25	57	65	57	57
j26	58	67	66	58
j27	63	64	60	57
j28	55	69	70	64
j29	64	69	63	53

j30	67	60	55	55
j31	68	69	60	67
j32	63	69	66	60
j33	57	62	67	62
j34	67	57	67	56
j35	67	58	68	56
j36	68	56	52	60
j37	56	65	70	68
j38	52	69	52	59
j39	54	62	51	52
j40	55	69	64	69
j41	50	52	64	57
j42	63	63	62	69
j43	67	54	68	61
j44	68	64	57	61
j45	58	67	57	53
j46	52	52	67	53
j47	63	62	55	60
j48	53	53	65	63
j49	52	54	53	69
j50	67	67	58	66

Table a(J,I) 'matrix of constraint coefficients A'

	i1	i2	i3	i4
j1	0	1	0	0
j2	1	1	1	1
j3	1	0	1	1
j4	0	0	1	1
j5	1	1	0	1
j6	0	0	0	0
j7	1	1	1	0
j8	1	1	0	0
j9	1	0	1	1
j10	0	1	0	1
j11	1	0	1	1
j12	1	0	0	0
j13	0	0	0	1
j14	1	1	0	0
j15	1	0	1	0

```
j16         1         0         0         1
j17         1         1         1         1
j18         1         0         1         1
j19         0         1         1         0
j20         1         1         0         0
j21         0         1         1         0
j22         1         1         0         1
j23         1         0         1         0
j24         0         0         0         0
j25         1         1         0         1
j26         0         1         1         1
j27         1         1         1         1
j28         1         1         0         1
j29         1         1         0         1
j30         1         1         0         0
j31         0         0         1         1
j32         1         0         1         1
j33         1         1         0         0
j34         0         0         0         1
j35         0         0         0         1
j36         1         1         1         1
j37         1         0         0         1
j38         1         1         1         0
j39         0         0         0         1
j40         1         0         0         1
j41         1         1         0         0
j42         0         0         1         1
j43         1         1         1         1
j44         1         1         1         0
j45         1         1         1         0
j46         1         1         1         1
j47         0         0         0         0
j48         1         1         1         1
j49         1         1         0         0
j50         0         0         1         0
```

B.3 Dataset of the 5kp40 binary problem

```
Table c(J,K) 'matrix of objective function coefficients C'
            k1        k2        k3        k4        k5
j1          3         10        4         9         10
```

j2	5	4	8	9	4
j3	5	5	6	6	7
j4	5	3	3	4	5
j5	8	2	2	9	7
j6	5	5	9	6	4
j7	9	5	3	6	7
j8	4	3	3	6	2
j9	4	2	8	3	4
j10	3	9	7	5	7
j11	7	9	8	5	9
j12	8	3	5	4	3
j13	6	9	7	6	9
j14	8	7	9	5	4
j15	4	5	4	6	6
j16	5	2	8	3	8
j17	5	5	6	5	2
j18	5	2	5	5	3
j19	3	7	5	8	7
j20	3	7	4	6	5
j21	9	5	10	6	3
j22	5	7	8	10	4
j23	9	7	8	6	9
j24	4	8	4	4	2
j25	4	4	10	7	4
j26	5	3	5	8	7
j27	2	7	4	5	6
j28	7	6	5	6	7
j29	9	6	9	8	3
j30	9	3	8	4	7
j31	3	8	10	4	10
j32	9	9	5	10	9
j33	5	3	7	5	3
j34	5	6	7	4	7
j35	7	6	7	4	8
j36	3	9	3	8	4
j37	8	2	7	4	4
j38	5	7	7	3	9
j39	8	10	9	8	5
j40	8	4	8	10	7

Table a(J,I) 'matrix of constraint coefficients A'

	i1	i2	i3	i4	i5
j1	285	153	237	204	217
j2	308	192	345	162	289
j3	124	150	72	154	298
j4	131	262	227	299	370
j5	290	130	245	255	155
j6	315	71	134	270	253
j7	101	179	359	213	325
j8	323	52	57	189	398
j9	252	244	186	146	358
j10	232	389	324	232	155
j11	370	382	220	270	194
j12	232	79	202	284	184
j13	265	183	199	277	146
j14	355	62	60	79	344
j15	141	80	161	68	208
j16	163	174	139	135	286
j17	152	371	215	208	148
j18	346	192	130	389	225
j19	397	305	386	124	143
j20	135	299	107	248	259
j21	305	178	303	121	239
j22	201	357	138	145	190
j23	75	234	155	212	156
j24	369	350	318	102	94
j25	390	109	276	287	300
j26	115	260	263	79	368
j27	378	66	226	116	150
j28	80	146	349	197	65
j29	380	144	323	266	385
j30	386	265	389	238	286
j31	63	148	98	245	235
j32	324	208	334	101	347
j33	163	251	399	85	222
j34	152	131	95	252	189
j35	94	341	125	250	215
j36	360	297	164	361	199
j37	111	140	135	195	240
j38	290	337	316	151	53

j39	187	305	185	238	352
j40	272	159	74	269	186

B.4 Dataset of the 6kp50 binary problem

Table c(J,K) 'matrix of objective function coefficients C'

	k1	k2	k3	k4	k5	k6
j1	2	2	5	3	0	1
j2	5	3	3	4	4	3
j3	0	3	0	1	0	2
j4	0	2	1	3	5	4
j5	5	4	5	4	3	4
j6	4	3	1	4	5	5
j7	2	0	1	3	3	5
j8	3	1	4	1	0	2
j9	2	2	2	1	5	5
j10	0	5	3	0	1	4
j11	1	4	4	3	1	2
j12	2	0	1	0	5	4
j13	5	4	1	2	1	3
j14	2	0	4	2	3	3
j15	1	4	2	4	1	2
j16	4	1	3	1	4	2
j17	2	1	2	4	4	2
j18	0	4	2	4	4	4
j19	4	4	3	3	0	4
j20	3	2	0	0	2	3
j21	1	3	2	5	1	1
j22	3	1	2	3	0	1
j23	4	1	1	3	0	1
j24	4	4	3	5	1	0
j25	4	5	4	2	4	2
j26	4	3	4	0	4	4
j27	1	4	1	3	1	2
j28	3	4	0	0	2	4
j29	2	4	1	0	4	1
j30	3	2	3	3	5	1
j31	5	4	0	2	4	0
j32	4	4	3	4	0	1
j33	3	2	1	2	5	2
j34	0	3	5	0	3	2

j35	3	1	4	3	3	1
j36	4	4	2	2	3	1
j37	2	2	1	3	1	2
j38	2	4	2	5	1	3
j39	5	5	3	0	4	1
j40	0	2	5	2	1	3
j41	4	0	5	1	1	3
j42	3	2	5	2	4	3
j43	4	2	4	5	4	5
j44	3	4	4	3	3	0
j45	1	4	2	4	3	3
j46	3	4	1	5	2	2
j47	1	3	2	2	5	2
j48	4	4	4	3	3	0
j49	1	1	4	3	4	2
j50	1	3	2	3	3	5

Table a(J,I) 'matrix of constraint coefficients A'

	i1	i2	i3	i4	i5	i6
j1	1	0	0	0	0	1
j2	0	1	1	0	0	1
j3	0	0	0	1	1	0
j4	1	1	0	0	1	1
j5	0	0	1	0	1	1
j6	1	1	1	1	0	0
j7	1	1	0	1	0	1
j8	0	1	0	1	0	1
j9	1	1	1	0	0	1
j10	0	0	1	1	0	0
j11	0	0	0	1	0	1
j12	0	1	1	1	1	0
j13	1	0	0	1	1	1
j14	1	1	1	0	1	1
j15	0	1	1	0	1	1
j16	0	1	1	1	1	0
j17	0	0	1	0	0	1
j18	0	1	1	0	0	0
j19	0	1	1	0	1	0
j20	0	0	0	1	0	1
j21	0	0	0	1	1	0
j22	0	0	0	1	0	1

j23	1	0	1	0	0	1
j24	0	0	0	0	1	0
j25	1	0	1	0	0	0
j26	0	0	1	0	0	1
j27	0	0	1	1	0	1
j28	0	1	1	1	1	1
j29	0	0	1	1	0	1
j30	0	0	0	1	1	1
j31	1	1	0	1	1	1
j32	1	0	1	1	1	0
j33	0	0	1	0	1	1
j34	1	0	1	1	1	1
j35	1	0	0	1	1	0
j36	0	0	1	1	0	0
j37	1	0	0	0	1	0
j38	0	1	1	0	1	0
j39	0	0	1	1	0	0
j40	0	0	0	1	0	0
j41	1	0	1	1	0	0
j42	1	0	0	1	0	1
j43	0	0	0	0	1	1
j44	1	1	0	0	0	1
j45	0	0	0	1	0	1
j46	1	0	1	1	0	1
j47	1	1	1	0	1	0
j48	0	1	1	1	1	0
j49	1	0	1	0	0	0
j50	0	1	1	0	1	0

References

Alves MJ, Costa JP (2009) An exact method for computing the nadir values in multiple objective linear programming. Eur J Oper Res 198(2):637–646

Arancibia AL, Marques GF, Mendes CAB (2016) Systems capacity expansion planning: Novel approach for environmental and energy policy change analysis. Environ Model Softw 85:70–79

Aras N, Yurdakul A (2016) A new multi-objective mathematical model for the high-level synthesis of integrated circuits. Appl Math Model 40(3):2274–2290

Attia AM, Ghaithan AM, Duffuaa SO (2019) a multi-objective optimization model for tactical planning of upstream oil & gas supply chains. Comput Chem Eng

Bababeik M, Khademi N, Chen A (2018) Increasing the resilience level of a vulnerable rail network: the strategy of location and allocation of emergency relief trains. Transp Res Part E Logist Transp Rev 119:110–128

Bal A, Satoglu SI (2018) A goal programming model for sustainable reverse logistics operations planning and an application. J Clean Prod 201:1081–1091
Behmanesh R, Zandieh M (2019) Surgical case scheduling problem with fuzzy surgery time: an advanced bi-objective ant system approach. Knowl Based Syst 186:104913
Ben-Tal A, Bertsimas D, Brown DB (2010) A soft robust model for optimization under ambiguity. Oper Res 58(4-PART-2):1220–1234
Bertsimas D, Brown DB (2009) Constructing uncertainty sets for robust linear optimization. Oper Res 57(6):1483–1495
Bertsimas D, Sim M (2004) The price of robustness. Oper Res 52(1):35–53
Bootaki B, Mahdavi I, Paydar MM (2014) A hybrid GA-AUGMECON method to solve a cubic cell formation problem considering different worker skills. Comput Ind Eng 75:31–40
Bootaki B, Mahdavi I,Paydar MM (2016) New criteria for configuration of cellular manufacturing considering product mix variation. Comput Ind Eng 98:413–426
Cambero C, Sowlati T (2016) Incorporating social benefits in multi-objective optimization of forest-based bioenergy and biofuel supply chains. Appl Energy 178:721–735
Cambero C, Sowlati T, Pavel M (2016) Economic and life cycle environmental optimization of forest-based biorefinery supply chains for bioenergy and biofuel production. Chem Eng Res Des 107:218–235
Canales-Bustos L, Santibañez-González E, Candia-Véjar A (2017) A multi-objective optimization model for the design of an effective decarbonized supply chain in mining. Int J Prod Econ 193:449–464
Carrizosa E, Guerrero V, Morales DR (2019) Visualization of complex dynamic datasets by means of mathematical optimization. Omega 86:125–136
Dabiri N, Tarokh MJ, Alinaghian M (2017) New mathematical model for the bi-objective inventory routing problem with a step cost function: a multi-objective particle swarm optimization solution approach. Appl Math Model 49:302–318
Domínguez-Ríos MÁ, Chicano F, Alba E, del Águila I, del Sagrado J (2019) Efficient anytime algorithms to solve the bi-objective Next Release Problem. J Syst Softw 156:217–231
Doukas H, Nikas A (2020) Decision support models in climate policy. Eur J Oper Res 280(1):1–24
Ehrenstein M, Wang CH, Guillén-Gosálbez G (2019) Strategic planning of supply chains considering extreme events: novel heuristic and application to the petrochemical industry. Comput Chem Eng 125:306–323
Ehrgott M, Ryan DM (2002) Constructing robust crew schedules with bicriteria optimization. J Multi-Criter Decis Anal 11(3):139–150
Florios K, Mavrotas G (2014) Generation of the exact Pareto set in multi-objective traveling salesman and set covering problems. Appl Math Comput 237:1–19
Forouli A, Doukas H, Nikas A, Sampedro J, Van de Ven DJ (2019a) Identifying optimal technological portfolios for European power generation towards climate change mitigation: a robust portfolio analysis approach. Util Policy 57:33–42
Forouli A, Gkonis N, Nikas A, Siskos E, Doukas H, Tourkolias C (2019b) Energy efficiency promotion in Greece in light of risk: evaluating policies as portfolio assets. Energy 170:818–831
Gavranis A, Kozanidis G (2017) Mixed integer biobjective quadratic programming for maximum-value minimum-variability fleet availability of a unit of mission aircraft. Comput Ind Eng 110:13–29
Habibi F, Barzinpour F, Sadjadi SJ (2019) A mathematical model for project scheduling and material ordering problem with sustainability considerations: a case study in Iran. Comput Ind Eng 128:690–710
Hombach LE, Walther G (2015) Pareto-efficient legal regulation of the (bio) fuel market using a bi-objective optimization model. Eur J Oper Res 245(1):286–295
Hwang CL, Paidy SR, Yoon K, Masud ASM (1980) Mathematical programming with multiple objectives: a tutorial. Comput Oper Res 7(1–2):5–31
Inghels D, Dullaert W, Bloemhof J (2016) A model for improving sustainable green waste recovery. Resour Conserv Recycl 110:61–73
Jabbarzadeh A, Azad N, Verma M (2019) An optimization approach to planning rail hazmat shipments in the presence of random disruptions. Omega
Jenkins PR, Lunday BJ, Robbins MJ (2019) Robust, multi-objective optimization for the military medical evacuation location-allocation problem. Omega, 102088.
Kadziński M, Labijak A, Napieraj M (2017a) Integrated framework for robustness analysis using ratio-based efficiency model with application to evaluation of Polish airports. Omega 67:1–18

Kadziński M, Tervonen T, Tomczyk MK, Dekker R (2017b) Evaluation of multi-objective optimization approaches for solving green supply chain design problems. Omega 68:168–184
Khalili-Damghani K, Amiri M (2012) Solving binary-state multi-objective reliability redundancy allocation series-parallel problem using efficient epsilon-constraint, multi-start partial bound enumeration algorithm, and DEA. Reliab Eng Syst Saf 103:35–44
Khalili-Damghani K, Abtahi AR, Tavana M (2013) A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems. Reliab Eng Syst Saf 111:58–75
Khalili-Damghani K, Tavana M, Sadi-Nezhad S (2012) An integrated multi-objective framework for solving multi-period project selection problems. Appl Math Comput 219(6):3122–3138
Laumanns M, Thiele L, Zitzler E (2006) An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. Eur J Oper Res 169(3):932–942
Liu S, Papageorgiou LG (2013) Multiobjective optimisation of production, distribution and capacity planning of global supply chains in the process industry. Omega 41(2):369–382
Martello S, Monaci M (2020) Algorithmic approaches to the multiple knapsack assignment problem. Omega 90:102004
Mastorakis K, Siskos E (2016) Value focused pharmaceutical strategy determination with multicriteria decision analysis techniques. Omega 59:84–96
Mavrotas G (2009) Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. Appl Math Comput 213(2):455–465
Mavrotas G, Florios K (2013) An improved version of the augmented ε-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems. Appl Math Comput 219(18):9652–9669
Mavrotas G, Figueira JR, Antoniadis A (2011) Using the idea of expanded core for the exact solution of bi-objective multi-dimensional knapsack problems. J Global Optim 49(4):589–606
Mavrotas G, Figueira JR, Siskos E (2015a) Robustness analysis methodology for multi-objective combinatorial optimization problems and application to project selection. Omega 52:142–155
Mavrotas G, Gakis N, Skoulaxinou S, Katsouros V, Georgopoulou E (2015b) Municipal solid waste management and energy production: consideration of external cost through multi-objective optimization and its effect on waste-to-energy solutions. Renew Sustain Energy Rev 51:1205–1222
Mavrotas G, Skoulaxinou S, Gakis N, Katsouros V, Georgopoulou E (2013) A multi-objective programming model for assessment the GHG emissions in MSW management. Waste Manag 33(9):1934–1949
Mohammadi M, Jula P, Tavakkoli-Moghaddam R (2019) Reliable single-allocation hub location problem with disruptions. Transp Res Part E Logist Transp Rev 123:90–120
Mohammadkhani N, Sedighizadeh M, Esmaili M (2018) Energy and emission management of CCHPs with electric and thermal energy storage and electric vehicle. Therm Sci Eng Progress 8:494–508
Mohammed AM, Duffuaa SO (2020) A tabu search based algorithm for the optimal design of multi-objective multi-product supply chain networks. Expert Syst Appl 140:112808
Mousazadeh M, Torabi SA, Pishvaee MS, Abolhassani F (2018) Accessible, stable, and equitable health service network redesign: a robust mixed possibilistic-flexible approach. Transp Res Part E Logist Transp Rev 111:113–129
Musavi M, Bozorgi-Amiri A (2017) A multi-objective sustainable hub location-scheduling problem for perishable food supply chain. Comput Ind Eng 113:766–778
Oke O, Siddiqui S (2015) Efficient automated schematic map drawing using multiobjective mixed integer programming. Comput Oper Res 61:1–17
Paul NR, Lunday BJ, Nurre SG (2017) A multiobjective, maximal conditional covering location problem applied to the relocation of hierarchical emergency response facilities. Omega 66:147–158
Qiu R, Zhang H, Gao X, Zhou X, Guo Z, Liao Q, Liang Y (2019) A multi-scenario and multi-objective scheduling optimization model for liquefied light hydrocarbon pipeline system. Chem Eng Res Des 141:566–579
Rabbani M, Saravi NA, Farrokhi-Asl H, Lim SFW, Tahaei Z (2018) Developing a sustainable supply chain optimization model for switchgrass-based bioenergy production: a case study. J Clean Prod 200:827–843
Rahimi Y, Torabi SA, Tavakkoli-Moghaddam R (2019) A new robust-possibilistic reliable hub protection model with elastic demands and backup hubs under risk. Eng Appl Artif Intell 86:68–82

Rayat F, Musavi M, Bozorgi-Amiri A (2017) Bi-objective reliable location-inventory-routing problem with partial backordering under disruption risks: a modified AMOSA approach. Appl Soft Comput 59:622–643

Razm S, Nickel S, Sahebi H (2019) A multi-objective mathematical model to redesign of global sustainable bioenergy supply network. Comput Chem Eng 128:1–20

Resat HG, Turkay M (2015) Design and operation of intermodal transportation network in the Marmara region of Turkey. Transp Res Part E Logist Transp Rev 83:16–33

Resat HG, Unsal B (2019) A novel multi-objective optimization approach for sustainable supply chain: a case study in packaging industry. Sustain Prod Consump 20:29–39

Roshan M, Tavakkoli-Moghaddam R, Rahimi Y (2019) A two-stage approach to agile pharmaceutical supply chain management with product substitutability in crises. Comput Chem Eng 127:200–217

Saedinia R, Vahdani B, Etebari F, Nadjafi BA (2019) Robust gasoline closed loop supply chain design with redistricting, service sharing and intra-district service transfer. Transp Res Part E Logist Transp Rev 123:121–141

Şakar CT, Köksalan M (2013) A stochastic programming approach to multicriteria portfolio optimization. J Glob Optim 57(2):299–314

Sazvar Z, Rahmani M, Govindan K (2018) A sustainable supply chain for organic, conventional agro-food products: The role of demand substitution, climate change and public health. J Clean Prod 194:564–583

Schaeffer SE, Cruz-Reyes L (2016) Static R&D project portfolio selection in public organizations. Decis Support Syst 84:53–63

Sedighizadeh M, Esmaili M, Mohammadkhani N (2018) Stochastic multi-objective energy management in residential microgrids with combined cooling, heating, and power units considering battery energy storage systems and plug-in hybrid electric vehicles. J Clean Prod 195:301–317

Shah R, Reed P (2011) Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems. Eur J Oper Res 211(3):466–479

Shekarian M, Nooraie SVR, Parast MM (2019) An examination of the impact of flexibility and agility on mitigating supply chain disruptions. Int J Prod Econ

Sylva J, Crema A (2007) A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. Eur J Oper Res 180(3):1011–1027

Tartibu LK, Sun BOHUA, Kaunda MAE (2015) Optimal design study of thermoacoustic regenerator with lexicographic optimization method. J Eng Des Technol 13(3):499–519

Torabi SA, Hamedi M, Ashayeri J (2013) A new optimization approach for nozzle selection and component allocation in multi-head beam-type SMD placement machines. J Manuf Syst 32(4):700–714

Vafaeenezhad T, Tavakkoli-Moghaddam R, Cheikhrouhou N (2019). Multi-objective mathematical modeling for sustainable supply chain management in the paper industry. Comput Ind Eng

Van de Ven DJ, Sampedro J, Johnson FX, Bailis R, Forouli A, Nikas A, Doukas H (2019) Integrated policy assessment and optimisation over multiple sustainable development goals in Eastern Africa. Environ Res Lett 14(9):094001

Vieira M, Pinto-Varela T, Barbosa-Póvoa AP (2017) Production and maintenance planning optimisation in biopharmaceutical processes under performance decay using a continuous-time formulation: A multi-objective approach. Comput Chem Eng 107:111–139

Wang S, Wang X, Yu J, Ma S, Liu M (2018) Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. J Clean Prod 193:424–440

Wiedemann P (1978) Planning with multiple objectives. Omega 6(5):427–432

Witting K, Ober-Blöbaum S, Dellnitz M (2013) A variational approach to define robustness for parametric multiobjective optimization problems. J Global Optim 57(2):331–345

Xidonas P, Mavrotas G, Psarras J (2010) Equity portfolio construction and selection using multiobjective mathematical programming. J Glob Optim 47(2):185–209

Xidonas P, Mavrotas G, Zopounidis C, Psarras J (2011) IPSSIS: An integrated multicriteria decision support system for equity portfolio construction and selection. Eur J Oper Res 210(2):398–409

Xin S, Liang Y, Zhou X, Li W, Zhang J, Song X, Zhang H (2019) A two-stage strategy for the pump optimal scheduling of refined products pipelines. Chem Eng Res Des 152:1–19

Xiong B, Chen H, An Q, Wu J (2019) A multi-objective distance friction minimization model for performance assessment through data envelopment analysis. Eur J Oper Res

Yu L, Zhang C, Yang H, Miao L (2018) Novel methods for resource allocation in humanitarian logistics considering human suffering. Comput Ind Eng 119:1–20

Zhang W, Reimann M (2014) A simple augmented∈-constraint method for multi-objective mathematical integer programming problems. Eur J Oper Res 234(1):15–24
Zhang Y, Masuku CM, Biegler LT (2019) An MPCC reactive distillation optimization model for multi-objective Fischer-Tropsch synthesis. Comput Aided Chem Eng 46:451–456
Zhou L, Geng N, Jiang Z, Wang X (2018) Multi-objective capacity allocation of hospital wards combining revenue and equity. Omega 81:220–233