Cooperative multiobjective optimization with bounds on objective functions

I. Kaliszewski^{1,2} · J. Miroforidis¹

Received: 30 January 2019 / Accepted: 31 August 2020 / Published online: 19 September 2020
© The Author(s) 2020

Abstract

When solving large-scale multiobjective optimization problems, solvers can get stuck because of memory and/or time limitations. In such cases, one is left with no information on the distance to the best feasible solution, found before the optimization process has stopped, to the true Pareto optimal solution. In this work, we show how to provide such information. To this aim we make use of the concept of lower shells and upper shells, developed in our earlier works. No specific assumptions about the problems to be solved are made. We illustrate the proposed approach on biobjective multidimensional knapsack problems derived from single-objective multidimensional knapsack problems in the Beasley OR Library. We address cases when a top-class commercial mixed-integer linear solver fails to provide Pareto optimal solutions attempted to be derived by scalarization.

Keywords Multiobjective optimization · Pareto optimality · Two-sided Pareto front approximations

1 Introduction

The ubiquitous access to more and more powerful computing platforms stimulates the appetite to solve more and more large optimization problems. However, various barriers (time and/or memory limits, “the curse of dimensionality” in combinatorial problems) are the cause that in practice one is often left with suboptimal solutions. In such cases, the existence of lower and upper bounds on the exact solution is essential. The multiobjective optimization literature is rich with various approaches to this issue, cf. [14, 15] for brief overviews.

This work has been inspired by the fact that when solving large-scale multiobjective optimization (MO) problems, it can happen that Pareto optimal (efficient) solutions are not derived. The reasons for that can be twofold. When making use of heuristics, such as Evolutionary Multiobjective Optimization (EMO), or other, Pareto optimality of derived solutions is not guaranteed. When making use of exact methods, such as mixed-integer programming

✉ J. Miroforidis
janusz.miroforidis@ibspan.waw.pl

¹ Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, Warsaw 01-447, Poland

² Warsaw School of Information Technology, ul. Newelska 6, Warsaw 01-447, Poland

(MIP), solvers can reach memory limits, and, less often, time limits, before the optimal solution is found. In both cases, one is left with a feasible solution which may be believed to be Pareto optimal or close to Pareto optimality, but no information is offered about how wide the actual gap is (*the Pareto suboptimality gap*) between the image (in the objective space) of the feasible solution derived and the image (in the objective space) of the set of Pareto optimal solutions (the Pareto front).

To cope with such unfavorable situations and to provide the lacking information, here we employ the general methodology developed for calculating bounds on objective function values of implicit Pareto optimal solutions [12,14,15,17,18]. The central concept of this methodology is to build two-sided approximations of the Pareto front (PF). To this aim, two sets of elements are produced, one composed of feasible solutions, called *lower shell*, and another composed of feasible or infeasible solutions, called *upper shell*. The images of these sets form, respectively, an internal and an external approximation of the PF. No element of the lower shell needs to be Pareto optimal.

Lower and upper shells which we make use of are composed of just elements (points, solutions). No relationship between them (in the form of constructing lines, triangles, polygons, etc.), except nondominancy, is assumed. Following the terminology introduced by Ruzika and Wiecek [28], they are 0-order approximations of PFs, whereas, e.g., the bound sets [8] or the attainment function [9] are of an order higher than 0.

In the literature, two general approaches to approximate the PF are present. The first approach, based on exact optimization methods, aims at building approximating constructs starting from a number of efficient solutions derived by an exact optimization method. From a vast literature on that approach one can quote [28] (a survey paper), [1,8,11] and [20].

The second approach, based on inexact, mostly population type optimization methods (cf., e.g., [2,4–6,10,25–27,29,30]), aims at producing discrete feasible approximations (lower shells) of Pareto fronts. In the second approach, no guarantee is offered that the resulting approximations include actual elements of the PF. In consequence, the actual behavior of such methods can be only observed on test problems with known sets of Pareto optimal solutions.

The methodology we refer to [12,15,17,24] follows the second approach. However, in contrast to other methods consistent with that approach, the said methodology provides for two-sided, lower and upper bounds on objective functions values of Pareto optimal solutions. By this, the behavior of inexact methods can be put under control. Moreover, responding to the needs of Multiple Criteria Decision Making, the methodology abstains from approximating the whole Pareto front and takes a more pragmatic course. Namely, it produces *ad hoc* local approximations of the Pareto front, only in regions of the explicit decision maker's interest. This provides for large savings by forgoing vain computations.

As the said methodology works only for a special case (see Sect. 4), in this work, we extend this methodology to the general MO setting.

The outline of the work is as follows. In Sect. 2, we recall the multiobjective problem formulation in its most general form and recall a method for Pareto optimal solution derivation, which is instrumental for the subsequent development. In Sect. 3, we present our development of general lower and upper bounds and in Sect. 4, we discuss a special case. Section 5 addresses the issue of derivation of upper shells. We illustrate the development in Sect. 6 on large-scale biobjective multidimensional knapsack problems. Section 7 concludes.

2 Preliminaries

Let x denote a solution, X a space of solutions, $X \subseteq \mathbb{R}^n$, X_0 a set of feasible solutions, $X_0 \subseteq X$. Then the general multiobjective optimization problem is defined as:

$$\begin{aligned} & \text{vmax} f(x) \\ & \text{subject to: } x \in X_0, \end{aligned} \quad (1)$$

where $f : X \rightarrow \mathbb{R}^k$, $f = (f_1, \dots, f_k)$, $f_l : \mathbb{R}^n \rightarrow \mathbb{R}$, $l = 1, \dots, k$, $k \geq 2$, are objective functions, and vmax denotes the operator of deriving all Pareto optimal solutions in X_0 . $\mathbb{R}^k$ is called the objective space.

Solution $\bar{x}$ is Pareto optimal (or: efficient) if $f_l(x) \geq f_l(\bar{x})$, $l = 1, \dots, k$, implies $f(x) = f(\bar{x})$. If $f_l(x) \geq f_l(\bar{x})$, $l = 1, \dots, k$, and $f(x) \neq f(\bar{x})$, then we say that x dominates $\bar{x}$ and we write $\bar{x} < x$. Solution $\bar{x}$ is weakly Pareto optimal (or: weakly efficient) if there is no such x that $f_l(x) > f_l(\bar{x})$, $l = 1, \dots, k$. Below, we shall denote the set of Pareto optimal solutions (or: efficient set) to (1) by N . Set $f(N)$ is called the Pareto front (PF).

It is a well-established result (cf. [7,12,23]) that solution x is Pareto optimal (actually, this solution is properly Pareto optimal, for a formal treatment of this issue cf., e.g., [12], pp. 36-37, [7], p. 121, [23], p. 103) if and only if it solves the Chebyshev weighted optimization problem

$$\min_{x \in X_0} \max_l [\lambda_l (y_l^* - f_l(x)) + \rho e^k (y^* - f(x))], \quad (2)$$

where weights $\lambda_l > 0$, $l = 1, \dots, k$, $e^k = (1, 1, \dots, 1)$, $y_l^* = \hat{y}_l + \varepsilon$, $\hat{y}_l = \max_{x \in X_0} f_l(x) < \infty$, $l = 1, \dots, k$, $\varepsilon > 0$ ($\hat{y}$ is called the ideal point), and ρ is a positive "sufficiently small" number. By solving problem (2) with $\rho = 0$ weakly Pareto optimal solutions are derived (cf., e.g., [23], p. 98).

By the "only if" part of this result, no Pareto optimal solution is a priori excluded from being derived by solving an instance of optimization problem (2). In contrast to that, maximization of a weighted sum of objective functions over X_0 does not possess, in general (and especially in the case of problems with discrete variables), this property (cf., e.g., [12], pp. 37-38, [19], pp. 54-56, [7], p. 121, [23], p. 78).

On the first glance, the objective function in (2) seems to be difficult to handle. However, problem (2) is equivalent to

$$\begin{aligned} & \min_{x \in X_0} s \\ & \text{subject to: } s \geq \lambda_l (y_l^* - f_l(x)) + \rho e^k (y^* - f(x)), \quad l = 1, \dots, k. \end{aligned} \quad (3)$$

In the sequel, we will assume that Pareto optimal solutions are derived by solving problem (2) with varying λ_l , $l = 1, \dots, k$.

3 Development of general lower and upper bounds

Given $\lambda = (\lambda_1, \dots, \lambda_k)$, let $x^{P_{opt}}(\lambda)$ denote a Pareto optimal solution to problem (2), which would be derived if this problem were solved to optimality (the implicit Pareto optimal solution). By the definition of $\hat{y}$, $f(x^{P_{opt}}(\lambda)) \in Y = \{y \in \mathbb{R}^k \mid y_l \leq \hat{y}_l, l = 1, \dots, k\}$.

The lower bound we developed in our earlier works [12] is general, i.e., it is valid for any problem of the form (1). Here, for completeness sake, we present the underlying arguments. To simplify the presentation, we assume here $\rho = 0$ (cf. [16], pp. 13-15, for the derivation of the formula for lower bounds with $\rho > 0$). In this case, $x^{Opt}(\lambda)$ can be weakly efficient.

To propose constructive, i.e., conjunctive bounds, one has to exploit the fact that Pareto optimal solutions are derived, as assumed, by solving problem (2).

Given multiple solutions of $\bar{x} \in X_0$, for each coordinate the maximal lower bound is selected. Solutions which are dominated by another elements of X_0 clearly provide lower

bounds which are lower than or equal to lower bounds provided by dominating solutions. Therefore, to avoid redundancy, it is reasonable to work with solutions which do not dominate one another and this is formalized by the notion of lower shell.

Lower shell to problem (1) is a finite nonempty set $S_L \subseteq X_0$, elements of which satisfy

$$\forall x \in S_L \nexists x' \in S_L \ x \prec x'. \quad (4)$$

The formula for lower bounds with $\rho \geq 0$ ([12], pp. 92–94, [16], pp. 13–15) is

$$\begin{aligned} f_l(x^{P_{opt}}(\lambda)) &\geq L_l(\lambda, S_L) \\ &:= \max\{\max_{x \in S_L}\{y_l^* - (\lambda_l + \rho)^{-1}[\max_{j=1,\dots,k} \lambda_j(y_j^* - f_j(x)) \\ &\quad + \rho e^k(y^* - f(x))]\} + \frac{\rho}{\lambda_l + \rho} \sum_{j \neq l}^k (y_j^* - U_j(\lambda))\}, \bar{L}_l\}, \quad l = 1, \dots, k, \end{aligned} \quad (5)$$

where $\bar{L}_l$ are lower bounds implied by some *a priori* information on $f_l(x^{P_{opt}}(\lambda))$, if available. If such lower bounds are not available, then one can simply set $\bar{L}_l = -\infty$. $U_l(\lambda)$ are known upper bounds on $f_l(x^{P_{opt}}(\lambda))$ (e.g., when one can infer these bounds basing on properties of problem (1)). If $U_l(\lambda)$ are not known, then one can simply set $U_l(\lambda) = y_l^*$.

3.2 Upper bounds

*Upper shell*¹ is a finite nonempty set $S_U \subseteq \mathbb{R}^n$, elements of which satisfy

$$\forall x \in S_U \nexists x' \in S_U \ x' \prec x, \quad (6)$$

$$\forall x \in S_U \nexists x' \in N \ x \prec x', \quad (7)$$

Not every element of an upper shell provides an upper bound on a component of $f(x^{P_{opt}}(\lambda))$. To be so, elements of upper shell have to be appropriately located with respect to a lower bound $L(\lambda, S_L)$.

Let S_U be an upper shell to problem (1).

Proposition 1 $x \in S_U$ provides an upper bound for some $f_l(x^{P_{opt}}(\lambda))$ only if for at least one l , $L_l(\lambda, S_L) \geq f_l(x)$.

Proof $x \in S_U$ provides no upper bound for $f_l(x^{P_{opt}}(\lambda))$ if for all l , $l = 1, \dots, k$, $L_l(\lambda, S_L) < f_l(x)$, as in this case nothing precludes $f_l(x^{P_{opt}}(\lambda)) \geq f_l(x)$ for one or several l . $\square$

Remark 1 It is worth observing that also some Pareto optimal solutions can satisfy conditions of Proposition 1. This relates to the development of [12] where lower and upper bounds were proposed with shells (subsets of Pareto optimal solutions), the concept in which lower and upper shells coincide. Later on, with applications to space sampling algorithms (e.g., evolutionary) in mind [14, 17, 24], to clearly delineate feasible and infeasible sampling spaces, we have adopted the condition $S_U \subseteq \mathbb{R}^n \setminus X_0$. However, for the sake of generality, here we stick to the definition of upper shell as that given above.

Proposition 2 Suppose $x \in S_U$ and $L_{\bar{l}}(\lambda, S_L) \leq f_{\bar{l}}(x)$ for some $\bar{l}$ and $L_l(\lambda, S_L) \geq f_l(x)$ for all $l = 1, \dots, k$, $l \neq \bar{l}$. Then x provides an upper bound for $f_{\bar{l}}(x^{P_{opt}}(\lambda))$, namely $f_{\bar{l}}(x^{P_{opt}}(\lambda)) \leq f_{\bar{l}}(x)$.

¹ In contrast to our earlier works [14, 17]), here we omit the requirement that elements of an upper shell should satisfy $f(x) \geq y^{nad}$, where y^{nad} is the nadir point. This specific and rather technical requirement was related to multiobjective evolutionary optimization context in which the idea of bounds was first proposed.

Fig. 2 $\bar{x} \in S_U$ provides a constructive upper bound for $f(x^{P_{opt}}(\lambda))$ (hypothetical location), namely $f_2(\bar{x})$, as indicated by $\circ$. Since $\bar{x} \in S_U$, $f(x^{P_{opt}}(\lambda))$ cannot belong to the dotted area

Proof By assumption, $f_l(x^{P_{opt}}(\lambda)) \geq L_l(\lambda, S_L) \geq f_l(x)$ for all $l = 1, \dots, k$, $l \neq \bar{l}$. Since $x \in S_U$ and hence x is not dominated by $f(x^{P_{opt}}(\lambda))$, the condition $f_{\bar{l}}(x^{P_{opt}}(\lambda)) \leq f_{\bar{l}}(x)$ must hold $\square$

4 The special case

Lower and upper bounds were originally developed ([12], pp. 92–102, [16], pp. 13–17) for the special case when at $x^{P_{opt}}(\lambda)$, the optimal solution to (3) (and hence, also to (2)), the following holds

$$s = \lambda_l(y_l^* - f_l(x^{P_{opt}}(\lambda))) + \rho e^k(y^* - f(x^{P_{opt}}(\lambda))), \quad l = 1, \dots, k. \quad (8)$$

In other words, $f(x^{P_{opt}}(\lambda))$ lies at the apex of the contour of the objective function of (2).

As said, lower bounds developed in three references cited above are valid for the general case, and only upper bounds are case specific.

Given $\bar{x} \in S_U$, by the definition of S_U , the part of the half line emanating from y^* (the locus of apexes of the objective function of (2)) belonging to $\{f(x) \mid f(x) \geq f(\bar{x})\}$ cannot contain $f(x^{P_{opt}}(\lambda))$. Thus, the intercept point of this half line with $\{f(x) \mid f(x) \geq f(\bar{x})\}$ defines upper bounds for some components of $f(x^{P_{opt}}(\lambda))$. For $k = 2$, this is illustrated in Fig. 2. The formula for upper bounds for this case can be found in three references cited above.

5 Derivation of upper shells

To make use of Proposition 2 we need S_U . Usually it is not known which part of $\mathbb{R}^n$ should be searched for elements of S_U . However, upper shells can be derived from some relaxations of problem (1), as shown by Proposition 3.

Let $X_0 \subset X'_0 \subseteq \mathbb{R}^n$. Consider the problem

$$\begin{aligned} & \text{vmax} f(x) \\ & \text{subject to: } x \in X'_0, \end{aligned} \quad (9)$$

with the set of efficient elements N' .

Proposition 3 Any subset of N' is an upper shell to problem 1.

Proof Since

$$\forall x \in N' \nexists x' \in N' \ x' \prec x,$$

for any $A \subseteq N'$ we have

$$\forall x \in A \nexists x' \in A \ x' \prec x,$$

thus A satisfies condition (6).

Since

$$X_0 \subset X',$$

we have

$$\forall x \in N' \nexists x' \in X_0 \ x \prec x'.$$

Hence,

$$\forall x \in N' \nexists x' \in N \ x \prec x',$$

and

$$\forall x \in A \nexists x' \in N \ x \prec x',$$

thus A satisfies condition (7).

In consequence, A is an upper shell. $\square$

Figure 3 gives a schematic illustration to Proposition 3. The significance of Proposition 3 is in that it paves a way to derive upper shells in an algorithmic way. Its practical usefulness depends on whether the relaxed problems can be solved to optimality in fractions of times spent on deriving incumbents of the original problems unsolved to optimality.

6 Numerical experiments

We apply the development of Sect. 5 to the case where large-scale multiobjective multi-dimensional knapsack problems solved by an optimization solver have been not solved to optimality. It is well known (see, e.g., [21]) that the knapsack problem is $\mathcal{NP}$ -hard. The multidimensional knapsack problem, being its generalization, is also $\mathcal{NP}$ -hard, so the multiobjective multidimensional knapsack problem is as well. It is also known that the difficulty of solving this problem by an MIP solver depends on the number of decision variables as well as the structure of the constraint set (see, e.g., [3]).

In this work, we say that the multiobjective multidimensional knapsack problem is a large-scale problem if it cannot be solved to optimality by a highly specialized MIP solver within a reasonable memory or time limit. We show how in such cases the lacking information about the size of the Pareto suboptimality gap can be retrieved.

Fig. 3 An illustration to Proposition 3—a case when $N' \neq N$

The multiobjective multidimensional knapsack problem is formulated as follows.

$$vmax \begin{cases} \sum_{j=1}^n c_{1,j} x_j \\ \dots \\ \sum_{j=1}^n c_{k,j} x_j \end{cases} \quad (10)$$

subject to: $x \in X_0$,

where $X_0 = \{x \mid \sum_{j=1}^n a_{i,j} x_j \leq b_i, i = 1, \dots, m, x_j \in \{0, 1\}, j = 1, \dots, n\}$, and all $a_{i,j}$, $c_{i,j}$ are nonnegative.

6.1 Solving multidimensional knapsack problems on the NEOS platform

To solve multiobjective mixed-integer linear problems (3) (weights λ_j are real numbers, hence variable s , the objective function in (3), assumes real values) we have to rely on MIP solvers. Our choice was CPLEX, because it is renown as an effective and robust MIP optimizer. Moreover, it is accessible free of charge, e.g., via NEOS platform (<https://neos-server.org/neos/solvers/lp:CPLEX/LP.html>). Running optimization problems on an open access platform ensures perfect result reproducibility. At the time of numerical experiments, this platform run CPLEX version 12.6.2.0. With NEOS, jobs are terminated whenever time limit of 8 hours of CPU or memory limit of 3 GB is reached. However, as NEOS reaching one of these limits provides no output log, we put the memory limit to 2.048 GB².

6.2 Multidimensional knapsack test problems

For test problems we selected single-objective multidimensional knapsack problems from Beasley OR Library (<http://people.brunel.ac.uk/mastijb/jeb/info.html>), collected in [3], and we extend them to biobjective problems (cf. Sect. 6.3). The library contains 270 problems grouped in 9 groups (the first number indicates the number of constraints, the second indicates the number of variables): (5, 100), (5, 250), (10, 100), (5, 500), (10, 250), (10, 500), (30,

² Advised by the NEOS support team, in all CPLEX runs we used the following CPLEX parameter settings: set workmem 128, set mip strategy file 2, set mip limits treememory 2048.

100), (30, 250), (30, 500). Each group contains three subgroups, each with different right hand side coefficients

$$b_i = \alpha \sum_{j=1}^n a_{i,j}$$

where α is the tightness ratio, equal to 0.25 for the first ten, 0.50 for the next ten, and 0.75 for the remaining ten problems in a group.

To prepare input data for CPLEX we converted all the 270 knapsack problems in the Beasley OR Library to the LP format. We made the converted problems publicly available via the web ³.

With CPLEX running on NEOS, we were able to solve to optimality almost all single-objective problems from (5, 500) and (30, 100) groups, a few problems from (10, 250) group and no problem from (10, 500), (30, 250) and (30, 500) groups. The detailed results are given in [13]. In each case, the suboptimality gap ⁴ is defined by the following formula

$$\text{subopt. gap} = 100 * \frac{\text{current MIP best bound} - \text{objective funct. value for incumbent}}{\text{objective funct. value for incumbent}},$$

and all those values are provided in CPLEX logs (where the suboptimality gap is termed *gap*).

6.3 Biobjective multidimensional knapsack problems

We generated biobjective multidimensional knapsack (BMK) problems (10) from Chu and Beasley problems. For each original Chu and Beasley problem we added the second objective function obtained by permuting the coefficients of the first.

To solve BMK problems constructed in that manner, we made use of transformation (3). In case of knapsack problems, the resulting problems are MIP problems, thus they can be handled by CPLEX.

To simplify modifications of the input LP files, reflecting varying weights λ_l , we introduced two unconstrained (free) variables f_1 and f_2 , equal to the first and the second objective function, respectively,

$$\begin{aligned} -f_1 + \sum_{j=1}^n c_{1,j} x_j &= 0, \\ -f_2 + \sum_{j=1}^n c_{2,j} x_j &= 0. \end{aligned}$$

³ http://www.ibspan.waw.pl/~kaliszew/MK_problems.

⁴ The term “optimality gap” is usually reserved for the difference between the optimal objective function value in the primal problem and in its dual.

Both variables f_1 and f_2 are clearly nonnegative. With that convention, the transformed problems have the following form, consistent with the LP format:

$$\begin{aligned} & \min s \\ \text{subject to: } & \begin{cases} s + (\lambda_1 + \rho)f_1 + \rho f_2 \geq \lambda_1 y_1^* + \rho(y_1^* + y_2^*), \\ s + \rho f_1 + (\lambda_2 + \rho)f_2 \geq \lambda_2 y_2^* + \rho(y_1^* + y_2^*), \\ -f_1 + \sum_{j=1}^n c_{1,j} x_j = 0, \\ -f_2 + \sum_{j=1}^n c_{2,j} x_j = 0, \\ \sum_{i,j} a_{i,j} x_j \leq b_i, \quad i = 1, \dots, m, \\ x_j \in \{0, 1\}. \end{cases} \end{aligned} \quad (11)$$

We set $\rho = 0.001$. In order to make y^* a relatively good approximation of $\hat{y}$, we maximized each objective function of problem (10) separately, setting the relative MIP gap tolerance to 1%. We set y_l^* to the optimal values of the respective objective functions derived with this tolerance. All these problems were solved in seconds. Obviously, $\hat{y}_l \leq y_l^*$. Hence, the use of y^* was legitimate.

We made all 270 transformed BMK problems (in the LP format) publicly available via the web⁵. For each problem in the set, λ_1 and λ_2 were set to default values 0.5. For different settings of λ_l , only the first two constraints in (11) have to be modified accordingly.

6.4 Solving biobjective multidimensional knapsack problems on the NEOS platform

We attempted to solve selected biobjective multidimensional problems (in the form (11) and with $\lambda_1 = \lambda_2 = 0.5$), namely the first, the eleventh and the twenty-first problem in group: (10, 500), (30, 100), (30, 250) and (30, 500), respectively. Given problem, let $x^{inc}(\lambda)$ denote an incumbent for a given λ . The results are given in Table 1. In all problems not solved to optimality the memory limit was reached.

We also attempted to solve one of these problems with as much memory as it was available on NEOS. We selected for that the first problem from the (10, 500) group and we set memory limit to 10 GB⁶. Even in this case the problem was not solved to optimality, but a better (in terms of the value of the objective function s) incumbent was found, see results in Table 2.

6.5 Bounds at work

In this section, we consider BMK problems, formulated as in (11), which we were not able to solve to optimality because CPLEX reached the memory limit. For example, this is the case of biobjective multidimensional knapsack problems from group (10, 500) of single-objective problems. It is the first group of BMK problems (counting from low dimensions up) in which we were not able to solve to optimality any BMK problem formulated as in (11) with CPLEX run on NEOS, because of the memory limit (we checked this for $\lambda_1 = \lambda_2 = 0.5$, cf. Sect. 6.4). Hence, for problems within this range of dimensions or higher, complementing CPLEX (or any other MIP solver) with a method to derive a kind of lower and upper bound information on Pareto optimal solutions is of utmost importance. However, having no access to CPLEX code, we can do this only in the form of an external procedure.

⁵ http://www.ibspan.waw.pl/~kaliszew/BMK_problems.

⁶ In this case the CPLEX parameters were: set workmem 128, set mip strategy file 2, set mip limits treememory 16384; cf. footnote in Sect. 6.2.

Table 1 Results of solving selected BMK problems; (o) denotes problems solved to optimality, x^{inc} time—conservative estimate of time to reach the incumbent, read from the CPLEX logs, $\lambda = (0.5, 0.5)$

Problem		l	$f(x^{inc}(\lambda))$	Solving time		$\frac{\text{Solving time}}{x^{inc} \text{ time}}$	
size	α			(sec.)	(tics)	(sec.)	(tics)
10,500	0.25	1	123163	294.53	250277.13	5.05	4.92
		2	124093				
10,500	0.50	1	220148	497.67	289376.71	1.03	1.02
		2	220933				
10,500	0.75	1	306520	460.21	238811.56	6.53	5.78
		2	306154				
(o) 30,100	0.25	1	21002	25.76	15109.89	4.84	4.83
		2	21542				
(o) 30,100	0.50	1	38694	369.21	306013.42	14.48	13.79
		2	38184				
(o) 30,100	0.75	1	53746	1.74	1064.38	19.33	35.86
		2	53721				
30,250	0.25	1	54587	2000.05	1534270.21	116.28	120.51
		2	54594				
30,250	0.50	1	103575	2206.10	1715527.50	2.73	2.74
		2	102980				
(o) 30,250	0.75	1	145535	2059.51	1733474.48	38.81	41.96
		2	144649				
30,500	0.25	1	113761	1692.08	853662.42	1.65	1.67
		2	113137				
30,500	0.50	1	212860	655.86	651950.40	3.90	3.87
		2	211188				
30,500	0.75	1	290954	531.31	360368.84	1.24	1.27
		2	290373				

Table 2 Results of solving the first problem from the (10, 500) group with the extended memory limit 10 GB; x^{inc} time—conservative estimate of time to reach the incumbent, read from the CPLEX logs, $\lambda = (0.5, 0.5)$

Problem		l	$f(x^{inc}(\lambda))$	Solving time		$\frac{\text{Solving time}}{x^{inc} \text{ time}}$	
size	α			(sec.)	(tics)	(sec.)	(tics)
10,500	0.25	1	123635	2431.40	1264404.10	1.41	1.42
		2	123627				

Moreover, it is well known that MIP solvers derive relatively quickly incumbents close to optimal solutions (in terms of the objective function values) or even optimal solutions but without the proof of optimality. Small incremental improvements or proving (by implicit enumeration) optimality of the incumbent usually consumes the majority of computing time. Existence of lower and upper bounds on objective function values of Pareto optimal solutions

allows one to decide if a satisfactory solution within an acceptability range has been already derived and, in consequence, allows to stop the optimization process early.

We applied lower bounds (formula (5)) and upper bounds (Proposition (2)) to selected BMK problems which were not solved on the NEOS platform to optimality because of the memory limit reached. In formula (5), we set $\bar{L}_l = 0$, (as the absolute lower bound on values of all objective functions of problem (10) is 0), $U_l(\lambda) = y_l^*$, $l = 1, 2$. The values of y_l^* for considered problems are shown in Table 3.

At first glance, it seems impossible to populate upper shells since the efficient set (N), the object to determined, is unknown. However, this can be done with the help of relaxations of problem (1) (see [15]). In this work, we take advantage of the fact that an MIP solver, when solving problem (11), solves also a series of its relaxations. To populate S_U , we used the MIP best bound on the objective function s (resulting from a problem relaxation), provided by CPLEX, and we derived corresponding values of objective functions f_1 and f_2 , denoted $f_1^{bb}(\lambda)$ and $f_2^{bb}(\lambda)$, by solving two equations (cf. (3))

$$\begin{aligned} s^{bb}(\lambda) &= \lambda_1(y_1^* - f_1^{bb}(\lambda)) + \rho e^k(y^* - f^{bb}(\lambda)), \\ s^{bb}(\lambda) &= \lambda_2(y_2^* - f_2^{bb}(\lambda)) + \rho e^k(y^* - f^{bb}(\lambda)), \end{aligned} \quad (12)$$

where $s^{bb}(\lambda)$ is the current MIP best bound and $f^{bb}(\lambda) = (f_1^{bb}(\lambda), f_2^{bb}(\lambda))$.

Remark 2 Let $x^{bb} \in \mathbb{R}^n$ be such that $f(x^{bb}) = f^{bb}(\lambda)$. No element of N dominates x^{bb} . Indeed, if an element of N dominated x^{bb} , then it is immediate to show that this element would produce s smaller than $s^{bb}(\lambda)$ corresponding to $f^{bb}(\lambda)$, a contradiction to x^{bb} defining the MIP best bound. If so, x^{bb} is a valid upper shell element.

At this point, we have no method to ensure condition (7) other than requesting elements x of an upper shell to satisfy $f_1^{bb}(\lambda) \leq f_1(x)$, $f_2^{bb}(\lambda) \leq f_2(x)$ with x^{bb} being clearly the best choice.

By Remark 2, it is guaranteed that x^{bb} is a legitimate element of an upper shell, e.g., one-element upper shell $\{x^{bb}\}$, but it may or may not satisfy conditions of Proposition 2. However, x^{bb} may satisfy these conditions for $x^{Popt}(\lambda')$ derived with $\lambda' \neq \lambda$. Along this line, for problems not solved to optimality for given $\tilde{\lambda}$ and conditions of Proposition 2 not satisfied, one should attempt to solve another two problems with λ' , λ'' , $\lambda'_1 > \tilde{\lambda}_1$, $\lambda'_2 < \tilde{\lambda}_2$ and $\lambda''_1 < \tilde{\lambda}_1$, $\lambda''_2 > \tilde{\lambda}_2$. These two problems, even if not solved to optimality, produce $(f_1^{bb}(\lambda'), f_2^{bb}(\lambda'))$ and $(f_1^{bb}(\lambda''), f_2^{bb}(\lambda''))$ which may yield valid upper bounds for $f(x^{Popt}(\lambda))$. If successful, these actions can be repeated with another two values of λ closer to $\tilde{\lambda}$, in the hope to get tighter upper bounds. However, since the problems being solved are discrete, there is a range of λ around $\tilde{\lambda}$ where all elements of the range yield the same $f(x^{Popt}(\lambda))$. Thus, there is a limit of upper bound tightness, specific for each individual problem.

It is clear that if $f_1^{bb}(\lambda')$ is an upper bound for $f_1(x^{Popt}(\lambda))$ and $f_2^{bb}(\lambda'')$ is an upper bound for $f_2(x^{Popt}(\lambda))$, then $f_1^{bb}(\lambda')$ is an upper bound for $f_1^{bb}(\lambda'')$ and $f_2^{bb}(\lambda'')$ is an upper bound for $f_2^{bb}(\lambda')$. However, the last observation does not apply to problems with $k \geq 3$.

We experimented with the first problems (formulated as in (11)) from (10, 500), (30, 250) and (30, 500) group (as seen in Table 1, the first problem from (30, 100) group was solved to optimality). The results of computations are presented in Table 4. Pareto sub-optimality gap is defined as vector $G_{Psub} = (G_{Psub,1}, \dots, G_{Psub,k})$, where $G_{Psub,l} = 100 * \frac{U_l(\lambda, S_U) - L_l(\lambda, S_L)}{U_l(\lambda, S_U)}$, $l = 1, \dots, k$.

For each problem in Table 4, lower shell S_L was expanded after each new incumbent was derived. After deriving $x^{inc}((0.5, 0.5))$, $S_L = \{x^{inc}((0.5, 0.5))\}$. After deriving

Table 3 Results of deriving y^* for selected BMK problems

Problem		l	y^*	Solving time	
size	α			(sec.)	(tics)
10,500	0.25	1	128522	0.11	58.42
		2	129424	0.08	56.70
30,250	0.25	1	58209	1.08	701.38
		2	58233	0.53	283.78
30,500	0.25	1	118686	0.15	85.4
		2	118028	0.17	75.76

The CPLEX relative MIP gap tolerance was set to 1%

$x^{inc}((0.45, 0.55))$, $S_L = S_L \cup \{x^{inc}((0.45, 0.55))\}$. After deriving $x^{inc}((0.55, 0.45))$, $S_L = S_L \cup \{x^{inc}((0.55, 0.45))\}$. We used this workflow (by changing λ vectors accordingly) to calculate bounds for the problem, as shown in Table 4 and Table 5.

As seen in Table 4, upper bounds on components of $f(x^{P_{opt}}(\lambda))$, $\lambda = (0.5, 0.5)$, are undefined for the first problem from group (30, 500). This is because elements of upper shells derived for $\lambda' = (0.45, 0.55)$ and $\lambda'' = (0.55, 0.45)$ do not satisfy conditions of Proposition 2. However, by taking another λ vectors, namely $\tilde{\lambda}' = (0.4, 0.6)$ and $\tilde{\lambda}'' = (0.6, 0.4)$, one can derive upper shells' elements being a valid source of upper bounds on components of $f(x^{P_{opt}}(\lambda))$ (see Table 5).

In Fig. 4 (for the first problem from group (30, 500)), one can see images (in the objective space) of incumbents (marked by rectangles) derived for set of lambda vectors $\Lambda = \{\lambda, \lambda', \lambda''\}$. Circles represent images of upper shells derived for set Λ . Filled rectangles represent images of incumbents derived for set of lambda vectors $\tilde{\Lambda} = \{\tilde{\lambda}', \tilde{\lambda}''\}$, and bullets—images of upper shells derived for set $\tilde{\Lambda}$. The bottom left corner of the dashed rectangle determines the vector of lower bounds on components of $f(x^{P_{opt}}(\lambda))$, and the upper right corner—the vector of upper bounds on components of $f(x^{P_{opt}}(\lambda))$. One can also see that the set of incumbents derived for $\lambda \in \Lambda \cup \tilde{\Lambda}$ forms a lower shell. The union of one-element upper shells derived for $\lambda \in \Lambda \cup \tilde{\Lambda}$ forms, in turn, an upper shell.

It is worth noting here that the "branch and bound"-based method for deriving approximations of the efficient set (hence—lower shells) of the multiobjective multidimensional knapsack problem presented in [22], also exploits the concept of relaxation to derive upper shells. However, in that approach, elements of an upper shell are used to control the derivation of elements of the efficient set approximation only.

7 Final remarks

When it comes to solving large-scale optimization problems, the concept of deriving the whole Pareto front loses its appeal and becomes just impractical. At most, the Pareto front can be navigated by solving instances of problem (2) with various λ (the method to interpret λ in decision making terms was given in [12,17,19]). Then, as suggested by Proposition 2, a number of upper shell elements can mutually and cooperatively provide upper bounds, and these bounds can be strengthened by deriving new upper shell elements.

In this work, we illustrated the proposed approach with pure combinatorial optimization problems. However, it can be applied to other class of problems, e.g., to mixed-integer quadratic programming problems. The cardinality constrained Markowitz portfolio selection

Table 4 Lower bounds, upper bounds and Pareto suboptimality gaps for selected BMK problems; x^{inc} time—conservative estimate of time to reach the incumbent, read from the CPLEX logs

Problem	α	l	λ	$f(x^{inc}(\lambda))$	$\bar{f}(\lambda)$	$L(\lambda, S_L)$	$U(\lambda, S_U)$	$G_{P_{sub}}$		Solving time		$\frac{\text{Solving time}}{x^{inc} \text{ time}}$	
								%		(sec.)	(tics)	(sec.)	(tics)
10,500	0.25	1	0.50	123163	123367.42	123152.36	123842.48	0.56		294.53	250277.13	5.05	4.92
		2	0.50	124093	124269.42	124054.36	124760.65	0.57					
		1	0.45	122541	122822.36	122530.45	123367.42	0.68		410.63	255547.23	21.12	20.11
		2	0.55	124688	124760.65	124519.85	—	—					
30,250	0.25	1	0.55	123730	123842.48	123657.07	—	—		420.73	256792.41	6.97	5.05
		2	0.45	123491	123704.59	123480.37	124269.42	0.63					
		1	0.50	54587	54864.53	54562.77	55171.74	1.10		2000.05	1534270.21	116.28	120.51
		2	0.50	54594	54888.53	54586.77	55199.85	1.11					
30,500	0.25	1	0.45	54364	54501.81	54174.88	54864.53	1.26		1770.39	1685251.21	32.96	33.10
		2	0.55	54938	55199.85	54931.02	—	—					
		1	0.55	54652	55174.74	54645.39	—	—		1697.75	1648936.54	4.43	9.76
		2	0.45	54590	54520.79	53879.23	55199.85	2.39					
30,500	0.25	1	0.50	113761	114275.67	113751.24	—	—		1692.08	855662.42	1.65	1.67
		2	0.50	113137	113617.67	113093.24	—	—					
		1	0.45	113224	113851.79	113214.14	114685.55	1.30		970.90	847766.47	1.53	1.54
		2	0.55	113580	114072.74	113549.22	—	—					
30,500	0.25	1	0.55	114133	114685.55	114122.99	—	—		1014.54	836961.20	1.42	1.42
		2	0.45	112511	113138.56	112453.23	114072.74	1.44					

Table 5 Lower bounds, upper bounds and Pareto suboptimality gaps for the selected problem of group (30, 500); x^{inc} time – conservative estimate of time to reach the incumbent, read from the CPLEX logs

Problem m,n	α	l	λ	$f(x^{inc}(\lambda))$	$\bar{f}(\lambda)$	$L(\lambda, S_L)$	$U(\lambda, S_U)$	G_{Psub}	Solving time		Solving time $\frac{x^{inc} \text{ time}}{x^{inc} \text{ time}}$	
									(sec.)	(tics)	(sec.)	(tics)
30,500	0.25	1	0.50	113761	114275.67	113751.24	115019.38	1.10	1692.08	853662.42	1.65	1.67
		2	0.50	113137	113617.67	113093.24	114402.80	1.14				
	0.40	1	0.40	112709	113248.20	112699.10	114257.67	1.38	768.56	854303.85	6.46	6.55
		2	0.60	114059	114402.80	114033.41	–	–				
	0.60	1	0.60	114888	115019.38	114547.58	–	–	964.02	840976.76	6.33	6.46
		2	0.40	111835	112528.08	111825.53	113617.67	1.58

Fig. 4 $\lambda = (0.5, 0.5)$. Element $f(x^{Pot}(\lambda))$ for the first problem of group (30, 500) is bounded by the left lower corner of the rectangle (the vector of lower bounds) and its right upper corner (the vector of upper bounds)

problem is an another example of a practical decision-making problem which large-scale instances can be treated with the help of the proposed approach.

The strength of the proposed development, as well as its weakness, lies in its generality (no assumption about problem (1) has been used). Though we present here experiments with biobjective problems, the very principle applies to any number of objectives. However, working efficiently with more than two objectives requires more sophisticated methods to populate upper shells. This will be the subject for further research.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Bradley, P.J.: Kriging-Pareto front approach for the multi-objective exploration of metamaterial topologies. *Prog. Electromag. Res. M* **39**, 141–150 (2014)
2. Deb, K.: *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester (2001)
3. Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. *J. Heuristics* **4**, 63–86 (1998)
4. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: *Evolutionary Algorithms for Solving Multi-objective Problems*. Kluwer Academic Publishers, New York (2002)
5. Coello Coello, C.A., Lamont, G.B. (eds.): *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific Printers, Singapore (2004)

6. Di Barba, P.: *Multiobjective Shape Design in Electricity and Magnetism*. Springer, Dordrecht (2010)
7. Ehrgott, M.: *Multicriteria Optimization*. Springer, Berlin (2005)
8. Ehrgott, M., Gandibleux, X.: Bound sets for biobjective combinatorial optimization problems. *Comput. Oper. Res.* **34**, 2674–2694 (2007)
9. Fonseca, C.M., Guerreiro, A.P., López-Ibáñez, M., Paquete, L.: On the computation of the empirical attainment function. In: Takahashi R.H.C., Deb K., Wanner E.F., Greco S. (eds) *Evolutionary Multi-Criterion Optimization. EMO 2011. Lecture Notes in Computer Science*, 6576, 106–120. Springer, Berlin, Heidelberg (2011)
10. Giri, B.K., Hakanen, J., Miettinen, K., Chakraborti, N.: Genetic programming through bi-objective genetic algorithms with study of a simulated moving bed process involving multiple objectives. *Appl. Soft Comput.* **13**, 2613–2623 (2013)
11. Hartikainen, M., Miettinen, K., Wiecek, M.: PAINT: Pareto front interpolation for nonlinear multiobjective optimization. *Comput. Optim. Appl.* **52**(3), 845–867 (2012)
12. Kaliszewski, I.: *Soft Computing for Complex Multiple Criteria Decision Making*. Springer, New York (2006)
13. Kaliszewski I.: Pareto suboptimal solutions to large-scale multiobjective multidimensional knapsack problems with assessments of Pareto suboptimality gaps (2017). [arXiv:1708.07069v2](https://arxiv.org/abs/1708.07069v2)
14. Kaliszewski, I., Miroforidis, J.: Two-sided pareto front approximations. *J. Optim. Theory Appl.* **162**, 845–855 (2014)
15. Kaliszewski, I., Miroforidis, J.: On upper approximations of Pareto fronts. *J. Glob. Optim.* **72**, 475–490 (2018)
16. Kaliszewski, I., Miroforidis, J., Podkopyayev, D.: Interactive multiple criteria decision making based on preference driven evolutionary multiobjective optimization with controllable accuracy—the case of ρ -efficiency. *Systems Research Institute Report*, RB/1/2011 (2011)
17. Kaliszewski, I., Miroforidis, J., Podkopaev, D.: Interactive multiple criteria decision making based on preference driven evolutionary multiobjective optimization with controllable accuracy. *Eur. J. Oper. Res.* **216**, 188–199 (2012)
18. Kaliszewski, I., Kiczowski, T., Miroforidis, J.: Mechanical design, multiple criteria decision making and Pareto optimality gap. *Eng. Comput.* **33**(3), 876–895 (2016)
19. Kaliszewski, I., Miroforidis, J., Podkopaev, D.: *Multiple Criteria Decision Making by Multiobjective Optimization: A Toolbox*. Springer, Berlin (2016a)
20. Lotov, A.V.: Decomposition of the problem of approximating the Edgeworth-Pareto hull. *Comput. Math. Math. Phys.* **55**(10), 1653–1664 (2015)
21. Martello, S., Toth, P.: *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Hoboken (1990)
22. Mavrotas, G., Florios, K., Figueira, J.R.: An improved version of a core based algorithm for the multi-objective multi-dimensional knapsack problem: a computational study and comparison with metaheuristics. *Appl. Math. Comput.* **270**, 25–43 (2015)
23. Miettinen, K.M.: *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Dordrecht (1999)
24. Miroforidis, J.: *Decision Making Aid for Operational Management of Department Stores with Multiple Criteria Optimization and Soft Computing*. PhD Thesis, Systems Research Institute, Warsaw (2010)
25. Osyczka, A.: *Evolutionary Algorithms for Single and Multicriteria Design Optimization*. Physica Verlag, Heidelberg (2002)
26. Qi, Y., Wu, F., Peng, X., Steuer, R.E.: Chinese corporate social responsibility by multiple objective portfolio selection and genetic algorithms. *J. Multi-Criteria Decis. Anal.* **20**, 127–139 (2013)
27. Ruiz, A.B., Luque, M., Ruiz, F., Saborido, R.: A combined interactive procedure using preference-based evolutionary multiobjective optimization. Application to the efficiency improvement of the auxiliary services of power plants. *Expert Syst. Appl.* **42**, 7466–7482 (2015)
28. Ruzika, S., Wiecek, M.M.: Approximation methods in multiobjective programming. *J. Optim. Theory Appl.* **126**, 473–501 (2005)
29. Talbi, E.: *Metaheuristics: From Design To Implementation*. Wiley, Hoboken (2009)
30. Tan, K.C., Khor, E.F., Lee, T.H.: *Multiobjective Evolutionary Algorithms and Applications*. Springer, Berlin (2005)