Computational Optimization and Applications (2022) 82:717–751
https://doi.org/10.1007/s10589-022-00379-7

Design of a heuristic algorithm for the generalized multi-objective set covering problem

Lakmali Weerasena[1] · Aniekan Ebiefung[1] · Anthony Skjellum[2]

Received: 29 April 2021 / Accepted: 12 May 2022 / Published online: 9 June 2022

Abstract

Set covering optimization problems (SCPs) are important and of broad interest because of their extensive applications in the real world. This study addresses the generalized multi-objective SCP (GMOSCP), which is an augmentation of the well-known multi-objective SCP problem. A mathematically driven heuristic algorithm, which uses a branching approach of the feasible region to approximate the Pareto set of the GMOSCP, is proposed. The algorithm consists of a number of components including an initial stage, a constructive stage, and an improvement stage. Each of these stages contributes significantly to the performance of the algorithm. In the initial stage, we use an achievement scalarization approach to scalarize the objective vector of the GMOSCP, which uses a reference point and a combination of weighted l_1 and l_∞ norms of the objective function vector. Uniformly distributed weight vectors, defined with respect to this reference point, support the constructive stage to produce more widely and uniformly distributed Pareto set approximations. The constructive stage identifies feasible solutions to the problem based on a lexicographic set of selection rules. The improvement stage reduces the total cost of selected feasible solutions, which benefits the convergence of the approximations. We propose multiple cost-efficient rules in the constructive stage and investigate how they affect approximating the Pareto set. We used a diverse set of GMOSCP instances with different parameter settings for the computational experiments.

✉ Lakmali Weerasena
Lakmali-Weerasena@utc.edu

Aniekan Ebiefung
Aniekan-Ebiefung@utc.edu

Anthony Skjellum
Tony-Skjellum@utc.edu

[1] Department of Mathematics, University of Tennessee at Chattanooga, 615 McCallie Avenue, Chattanooga, TN 37403-2598, USA

[2] SIM Center, University of Tennessee at Chattanooga, 615 McCallie Avenue, Chattanooga, TN 37403-2598, USA

Keywords Approximation · Algorithm · SCP · Multi-objective · Multi-cover

1 Introduction

Set Covering Problems (SCPs) naturally arise alongside diverse human activities in conservation biology, environmental science, computer science, and management science. They have contributed to an increased need for combinatorial mathematical optimization of generalized covering models. An SCP instance includes a finite set of items that are grouped into a family of subsets such that each item in the set is to be included in at least one subset of the family. The goal of the single-objective SCP (SOSCP) is to determine a subset of sets from the family, including each item in at least one set such that the total cost associated with the selected sets is minimized. Numerous studies have now been developed to solve the SOSCP.

Since SCP applications with multiple conflicting objective functions are more realistic, Jaszkiewicz [21] in 2003 introduced a generalization to the SOSCP by adding multiple conflicting linear objective functions. This extended problem is known as the multi-objective SCP (MOSCP). The optimality concepts in multi-objective optimization problems (MOPs) view competing solutions in terms of trade-offs concerning the multiple conflicting objectives rather than in terms of a clearly defined hierarchy of preferences between solutions, as implied in single-objective optimization. Incompatibility between any two solutions in multi-objective optimization is typical. Therefore, the challenge of verifying optimality within a single-objective environment due to the solution set's combinatorial structure increases when lifted into a multi-objective context. The MOSCP is one such example of note, and even its single objective counterpart is known to be NP-hard [23] as a combinatorial optimization problem.

In a multi-objective context, the concept of "optimum" has to be revisited. Instead of providing only one optimal solution, the approaches applied to MOPs should produce a set of solutions, known as Pareto optimal solutions. The Pareto optimal solutions form the Pareto set [8]. These solutions are optimal because there is no other better solution when all objective functions are taken into consideration. However, computing the exact Pareto set requires substantial computational effort to solve large-scale MOPs. Hence, approximation algorithms such as local search algorithms [1, 36], variable neighborhood search algorithms [27, 28], colony optimization algorithms [14, 24] are often used to estimate the Pareto sets of the MOPs. The approximations are commonly referred to as nondominated sets. A survey of such methods before the year 2000 can be found in [7, 9, 42]. Approximating the Pareto set of the MOSCP has received growing interest in the last two decades. In 2003 and 2004, Jaszkiewicz [21, 22] proposed the Pareto memetic algorithm (PMA) for multi-objective combinatorial optimization (MOCO) problems and used the MOSCP as a benchmark MOCO problem to analyze the performance of the PMA. In 2006, Prins et al. [37] proposed a two-phase algorithm to approximate the Pareto set of the bi-objective SCP. In 2011, Lust et al. [28] proposed a very large-scale neighborhood search method for solving MOCO problems and used the MOSCP as a benchmark problem to evaluate their performance. In 2014, Lust and Tuyttens [29]

applied an improved version of the very large-scale neighborhood search method to the MOSCP. In 2014, Florios and Mavrotas [12] proposed an exact algorithm to compute the Pareto sets of multi-objective integer programs and used the MOSCP to verify the accuracy of the algorithm. In 2017, Weerasena et al. [47] proposed a heuristic algorithm that partitions the feasible region of the MOSCP by adding branching constraints to produce subproblems of the MOSCP and approximating the Pareto set of the MOSCP with the help of the subproblems. In 2018, Weerasena and Wiecek [46] proposed an algorithm called the ϵ-approximation algorithm to approximate the MOSCP with a mathematically proven error bound.

Although the MOSCP focuses on covering each item at least once, the real-world applications of the SCP—such as vehicle routing, crew scheduling, and logical analysis of data [2, 3, 6, 16, 25, 30–32, 34, 38, 39, 45]—requires more than one set to cover each item (multiple covers). These problems are formulated as SOSCPs with generalized coverage constraints. In 2020, Weerasena [44] introduced a generalized MOSCP (GMOSCP) by adding generalized coverage constraints to the MOSCP, and proposed a heuristic algorithm to solve a reserve site selection problem in conservation biology with appropriately defined objective functions and coverage constraints.

In this study, we focus on approximating the Pareto set of the GMOSCP. The generalized coverage constraints and the multiple conflicting objective functions add an extra level of difficulty to this task. We propose an algorithm that depends on constructing and solving the GMOSCP subproblems through branching of the feasible region. The algorithm can be explained using a tree structure, and each node of the tree refers to a subproblem of the GMOSCP. The branching-algorithm inspired the development of our algorithm to solve single-objective mixed-integer optimization problems presented in [11]. This algorithm was applied to approximate the Pareto set of bi-objective mixed 0–1 integer linear programming problems in [40]. This algorithm is also successfully integrated into the algorithm proposed in [47] to approximate the Pareto set of the MOSCP. Our study presents a novel mathematically-driven heuristic algorithm to approximate the Pareto set of the GMOSCP by exploring the subproblems described by each node of the tree. Our search method identifies solutions established on two lexicographic rules for obtaining feasible solutions, and it is called the Lexicographic-Order Local Search (LOLS) algorithm.

In this paper, we argue that the number of sets needed to cover each item must be also be taken into consideration when the set selection rules are defined for the GMOSCP, since each item has multiple coverage requirements. This is the distinguished difference between the MOSCP approximation algorithms and the GMOSCP approximation algorithms. Our two lexicographic set rules are specially designed for this generalized case. The algorithm has three components, (1) initial stage, (2) constructive stage, and (3) improvement stage. The algorithm starts with a set of feasible solutions provided by the initial stage. The typical approach in obtaining solutions for the MOPS involves using certain replacement scalarizing functions that may depend on the objective values and also on some additional parameters.

In this study, we use achievement scalarization functions (ASF) proposed in [49], using a reference point and composite weighted l_1 and l_∞ norms of the objective functions to scalarize the objective weight of the GMOSCP at the initial stage. We consider several scalarizations of the objective function vector using uniformly

distributed weight vectors based on scalarizing achievement functions [10] concerning one reference point. We then solve the corresponding single-objective optimization problem, which is a common approach used in many earlier proposed algorithms [5, 13, 18, 20, 47] to obtain an initial feasible solution set. The advantage of using the reference points scalarization functions approach is that it benefits large-scale MOPs when approximating the Pareto sets in a parallel algorithmic framework [10].

In the constructive stage, we apply the local branching constraints to construct subproblems as in [11]. The branching constraints are defined based on Hamming distance [17], which is the absolute difference between a preferred solution $\bar{x}$ and a candidate solution x. Given the Hamming distance search length L, we add the branching constraints $N(\bar{x}, x) \leq L$ and $N(\bar{x}, x) > L$, where N denotes the search space to partition the feasible region. The branching constraint $N(\bar{x}, x) \leq L$ with the regular GMOSCP constraints define subproblems of the GMOSCP. If the search space of the subproblem is not examined intensively, the opportunity of finding better solutions to the GMOSCP can be lost. To overcome such issues, our algorithm will ensure an extensive examination of each search space of the subproblem. Each search space is extensively searched for new solutions by performing several searches. The search region with $N(\bar{x}, x) > L$ along with the regular GMOSCP constraints continued to be partitioned with respect to newly identified solutions. The partitioning of the feasible region of the GMOSCP using the branching constraints can be explained using tree-search strategy [11]. In the improvement stage, we examine each feasible solution further and improve the value of the objective vector by removing redundant sets.

In [26], the authors argued that due to the bias introduced by the cost-efficient rule of the SOSCP, the feasible solutions may converge to a local optimum instead of a global optimum in some problem instances; altering the cost-efficient rules is a meaningful strategy to reach the global optima in SOSCP. These authors used the cost-efficient rule defined in [4], which identifies the best sets based on a ratio determined by the cost over the number of uncovered items of sets. It is proven that the Chvátal's [4] greedy algorithm obtains a feasible solution within $\log m$ error terms, where m is the number of items in the test problem. Since the results yielded by this algorithm are very close to the optimal solutions, the study in [26] introduced slight modifications to Chvátal's [4] cost-efficient rule, which is also a motivation from [43], and showed that their heuristic best performs on the SOSCP among the heuristics based on the solution quality. Encouraged by this research on the SOSCP, we extend some of these rules for the GMOSCP and investigate their effects when approximating the Pareto set. We discover that having reasonably well multiple priority rules in the LOLS algorithm results in a better approximation for the GMOSCP. Our experimental analysis shows that all components of the algorithm are important for the success of the algorithm. For example, our results indicate that the two lexicographic set selection rules yield a significant improvement in the approximation quality. Besides, the experimental analysis gives insight into specific components' behavior, such as increasing either the number of scalarizations or the number of searches of the subproblems.

We have used a diverse set of GMOSCP instances with different parameter settings for the computational experiments. The GMOSCP can be reduced to an MOSCP by setting the number of sets needed to cover each item to one. Thus, the LOLS algorithm can be used to approximate the Pareto set of the MOSCP. We also analyzed the effectiveness of the LOLS algorithm with existing MOSCP algorithms. Our experimental comparison of the LOLS algorithm with the LB-AI algorithm shows that LOLS performance is very competitive or often superior on benchmark test problems.

The rest of this paper is arranged as follows: Sect. 2 describes the preliminaries on MOPs and the problem formulations. This section also provides the weighted-sum function and achievement scalarization function scalarization approaches of MOPs needed for our study. Section 3 provides the multiple cost-efficient rules and the approximation algorithm based on lexicographic set selection-rules. Section 4 provides the experimental setups, performance metrics, and test instances. Section 5 provides the results of the empirical analysis. We conclude the paper in Sect. 6.

2 Preliminaries and related background

Let $\mathcal{R}^p$ be a p Euclidean vector space. For two vectors $y^1, y^2 \in \mathcal{R}^p$, we write $y^1 \leqq y^2$ if $y^1_q \leq y^2_q$ for $q = 1, \ldots, p$; $y^1 \leq y^2$ if $y^1_q \leq y^2_q$ for all $q = 1, 2, \ldots, p$ and $y^1 \neq y^2$; and $y^1 < y^2$ if $y^1_q < y^2_q$ for all $q = 1, 2, \ldots, p$. The nonnegative orthant of $\mathcal{R}^p$ is defined as $\mathcal{R}^p_{\geqq} = \{y \in \mathcal{R}^p : y \geq \underline{0}\}$.

Let $f : \mathcal{R}^n \to \mathcal{R}^p$ be an objective vector and let $Q = \{q : q = 1, 2, \ldots, p\}$ be an index set. A multi-objective optimization problem (MOP) is defined as below.

$$\min f(x) = (f_1, \ldots, f_p) \quad \text{subject to } x \in X, \tag{1}$$

where $f_q : \mathcal{R}^n \to \mathcal{R}$ for $q \in Q$ is a scalar-valued function and x is feasible solutions in the the feasible set X. The outcome space Y is obtained by evaluating the p objective functions for $x \in X$. That is, $Y := f(X) \subset \mathcal{R}^p$.

The commonly used partial order optimality concept in multi-objective optimization is Pareto efficiency (or simply efficiency). A solution $x^* \in X$ is called an efficient solution for the MOP if there does not exist $x \in X$ such that $f(x) \leq f(x^*)$. The image $f(x) \in Y$ of an efficient solution is called a Pareto outcome. We denote the set of all efficient solutions by X_E. The image of X_E is denoted by $Y_p = f(X_E)$ and is referred to as the Pareto set [8]. A comparison of vectors can be used to check domination between any two elements in $Y \subset \mathcal{R}^p$.

Definition 1 If $x_1, x_2 \in X$ and $f(x_1) \leq fx_2)$, then x_1 is said to dominate x_2 and $y_1 = f(x_1)$ is said to dominate $y_2 = f(x_2)$. An element $y^* \in Y$ is called a nondominated point of Y if there does not exist $y \in Y$ dominating y^*.

The ideal objective vector of an MOP can be defined as follows.

Definition 2 The ideal objective vector $f^{id} = (f_1^*, \ldots, f_p^*)$ is computed by

$$f_q^* = \min_{x \in X} f_q(x),\ q \in Q \tag{2}$$

Below, we define the GMOSCP.

2.1 Generalized multi-objective set covering problem

The following notations are common in related SCP studies, and we use them throughout this text.

1. The set of m items $E = \{e_1, e_2, \ldots, e_m\}$ with the index set $I = \{i : i = 1, 2, \ldots, m\}$
2. A family of n subsets of E, $S = \{S_1, S_2, \ldots, S_n\}$ with the index set $J = \{j : j = 1, 2, \ldots, n\}$,

The GMOSCP is conceptually defined here as in Weerasena [44]. Let $a_{ij} = 1$ if $e_i \in S_j$ holds and $a_{ij} = 0$, otherwise. Let $x \in \{0, 1\}^n$ be the decision variable, such that $x_j = 1$ if S_j is selected to cover an item and $x_j = 0$ otherwise. Symbolically, the GMOSCP is defined as below.

$$\min f(x) = \left[f_1 = \sum_{j \in J} c_j^1 x_j, \ldots, f_p = \sum_{j \in J} c_j^p x_j \right] \text{ subject to } x \in X \tag{3}$$

where c_j^q denote the cost of set S_j regarding the objective function q for $q \in Q$, f_q is a real-valued linear function such that $f_q : \{0, 1\}^n \to \mathcal{R}$ and

$$X = \left\{ x \in \{0, 1\}^n : \sum_{j \in J} a_{ij} x_j \geq b_i \text{ for } i \in I, \text{where } b_i > 1 \text{ for at least one } i \in I \right\}. \tag{4}$$

The distinguishable difference between the MOSCP and the GMOSCP is the multi-cover constraint in which the coverage requirement is greater than one for at least one item. Using matrix notation, the GMOSCP can be written as

$$\min\ Cx \text{ subject to } Ax \geqq b \tag{5}$$

where $A \in \{0, 1\}^{m \times n}$ is a matrix such that a_{ij} denotes the entry at the i^{th} row and j^{th} column of a matrix A, $C \in \mathcal{R}^{p \times n}$ is a matrix such that c_j^q denotes the element at the q^{th} row and j^{th} column of the matrix C.

2.2 Exact methods

The approximation algorithm proposed in this study is developed based on the scalarizations of the objective function vector with respect to the Weighted-Sum Method and the Achievement Scalarizing Method, which are exact methods for finding efficient solutions of MOPs.

2.2.1 Weighted-sum scalarization

Let $\lambda = (\lambda_1, \ldots, \lambda_p)^T \in \mathcal{R}^p_{\geq}$ be a weight vector containing weighting coefficients of the objective functions and, typically, λ is normalized such that $\sum_{q \in Q} \lambda_q = 1$. The scalarized GMOSCP with the weighted-sum scalarization with respect to the weight vector λ can be written as follows:

$$\begin{aligned} \min f_{ws}(\lambda) = & \sum_{q \in Q} \lambda_q f_q \\ \text{subject to } & x \in X. \end{aligned} \tag{6}$$

where $f_{ws}(\lambda)$ denotes the scalarized weighted-sum objective function.

We obtain the relaxed weighted-sum problem in (6) by replacing the binary variables x_j for $j \in J$ with nonnegative variables $x_j \geq 0$ for $j \in J$ as shown in (7)

$$\begin{aligned} \min f_{rws}(\lambda) = & \sum_{q \in Q} \lambda_q f_q \\ \text{subject to } & x \in \bar{X} \end{aligned} \tag{7}$$

where $f_{rws}(\lambda)$ denotes the relaxed weighted-sum objective function and

$$\bar{X} = \left\{ x \in \mathcal{R}^n : \sum_{j \in J} a_{ij} x_j \geq b_i \text{ for } i \in I, \text{where } b_i > 1 \text{ for at least one } i \in I \right\}. \tag{8}$$

2.2.2 Achievement scalarization function

The technique of ASFs [48] performs well with reference points [10] that represent acceptable values for the objective functions. Let $F(f|f^0, \lambda, \rho) : Y \to \mathcal{R}$ be a function, $f = (f_1, f_2, \ldots, f_p)^T$ be an objective vector and $f^0 = (f^0_1, f^0_2, \ldots, f^0_p)^T \in Y$ be a reference point. The ASF, which is a combination of weighted l_1 and l_∞ norms of the objective function vector, is defined as

$$F(f|f^0, \lambda, \rho) = \max_{q \in Q} \left\{ \lambda_q (f_q - f^0_q) \right\} + \rho \sum_{q \in Q} \lambda_q (f_q - f^0_q) \tag{9}$$

where $\lambda = (\lambda_1, \ldots, \lambda_p) \in \mathcal{R}^p_{\geq}$ is a vector containing weighting coefficients of the objective functions and ρ is an arbitrary small positive number. We define the corresponding achievement optimization problem for the GMOSCP as

$$\begin{aligned} \min F(f|f^0, \lambda, \rho) = & \max_{q \in Q} \left\{ \lambda_q (f_q - f^0_q) \right\} + \rho \sum_{q \in Q} \lambda_q (f_q - f^0_q) \\ & x \in X \end{aligned} \tag{10}$$

Proposition 2.1 *If f^0 is a reference point, then the model (10) provides an efficient solution x^* for a given weight vector λ, and if x^* is an efficient solution, then there exists a function $F(f|f^0, \lambda, \rho)$ [41].*

To find the optimal solution of the model in (10), we optimize the following equivalent problem.

$$\begin{aligned} &\min \theta \\ &\text{subject to } \lambda_q(f_q - f_q^0) + \rho \sum_{q \in Q} \lambda_q(f_q - f_q^0) \leq \theta \; \forall \; q \in Q \\ &\quad x \in X \text{ and } \theta \text{ is a free variable} \end{aligned} \tag{11}$$

Using different weight vectors λ for the same reference point f^0 leads to different optimal solutions for the achievement problem defined in (11). It is important to mention that this model has p number of additional constraints and one additional free variable compared to the classical weighted-sum method discussed in Sect. 2.2.1. Next we discuss the design of the algorithm.

3 Design of the approximation algorithm

In this section we present our definitions of lexicographic set selection rules and multiple cost-efficient rules.

Definition 3 A vector $\bar{x} \in \{0, 1\}^n$ is called a partial solution if $\bar{x} \notin X$.

The proposed algorithm adds sets to convert the partial solution $\bar{x} \in \{0, 1\}^n$ to a feasible solution $\bar{x} \in X$. The sets are identified using two priority preferences: (1) cost-efficient preference and (2) coverage-efficient preference. We use these two preferences according to the lexicographic order preference.

3.1 Lexicographic set selection rules

Let $\bar{x}$ be a partial solution and $\bar{b} = (\bar{b}_1, \ldots, \bar{b}_m)^T$ be the corresponding coverage. The additional number of sets needed to obtain a feasible solution from $\bar{x}$ is given by $b^T - \bar{b}^T = (b_1 - \bar{b}_1, \ldots, b_m - \bar{b}_m)^T$. An item e_i for $i \in I$ is said to be covered if $b_i - \bar{b}_i \leq 0$. Let $\bar{J}$ contain the indices of the sets selected for the partial solution $\bar{x}$ and $\bar{I}$ contain the indices of the covered items by the partial solution $\bar{x}$, respectively. We use Definitions 4 to 6 to define our lexicographic rules.

Definition 4 (Set Density) Let S_j be a set for $j \in J \setminus \bar{J}$. The set density of S_j, denoted s_j^d, is defined as the number of uncovered items in the set S_j. That is,

$$s_j^d = |\{i \in I \setminus \bar{I} : a_{ij} = 1\}|.$$

The set density vector is denoted by $s^d = (s_1^d, s_2^d, \ldots, s_n^d)^T \in \mathcal{Z}^n$. The set density s_j^d is zero if the set S_j is selected for a solution.

When selecting additional sets to cover uncovered items in SCPs, the cost of a set and the number covered by the set play major roles. Since our main goal is to minimize the total cost of selected sites, cost-efficient approaches should give a higher priority to the cost coefficients c_j^q for $q \in q$ and $j \in J$ of the GMOSCP. Further, the set density defined in Definition 4 is related to the number of items in the set, but it is a dynamic factor for our algorithm as it changes from iteration to iteration. We describe this fact later in more detail. The multiple cost-efficient rules introduced for the SOSCP in [26] are functions of the cost coefficients and the items contained in the sets. Motivated by the [26] study, we define four different cost-efficient rules for the GMOSCP as functions of cost coefficients and set densities.

Definition 5 Let $\lambda \in \mathcal{R}_{\geq}^p$ be a weight vector. Let $\sum_{q\in Q} \lambda_q c_j^q$ and s_j^d be the scalarized cost and the set density of the set S_j, respectively. The four cost-efficient rules are denoted as $g_1(\lambda, c_j)$, $g_2(\lambda, c_j)$, $g_3(\lambda, c_j)$ and $g_4(\lambda, c_j)$ where,

$$g_1(\lambda, c_j) = \frac{\sum_{q\in Q} \lambda_q c_j^q}{s_j^d}, \; g_2(\lambda, c_j) = \frac{\sum_{q\in Q} \lambda_q c_j^q}{(s_j^d)^2},$$

$$g_3(\lambda, c_j) = \frac{\sqrt{\sum_{q\in Q} \lambda_q c_j^q}}{s_j^d}, \text{ and } g_4(\lambda, c_j) = \frac{\sum_{q\in Q} \lambda_q c_j^q}{\sqrt{s_j^d}}.$$

The classical cost-efficient rule introduced by Chvátal et al [4] for the SOSCP identifies the best sets based on a ratio determined by the cost over the number of uncovered items of sets. Thus, the ratio is calculated by a fractional term in which the denominator and numerator consist of linear terms. Motivated by the multiple cost-efficient rules proposed in [26], we investigate whether there is a better relationship between cost-efficient value and set density other than this linear relationship for the GMOSCP. We have designed these rules to investigate the best relationship between the cost-value and the set density. The cost-efficient rule g_1 is a linear fractional rule while the cost-efficient rules g_2, g_3, g_4 are non-linear fractional rules. The rule g_1 considers the linear combination of all cost values over the set-density. In rules, g_2, g_3, g_4, we have non-linear terms either in the numerator or the denominator. With the cost-efficient rule g_4, we prioritize the cost-value considering a non-linear fractional term. Thus, in contrast to cost-efficient rules g_1, g_2, g_3, g_4 enable us to enables us to provide more priority for the cost-efficient value. We analyze their effects in Sect. 5.

Since the GMOSCP requires different multiple coverage for each item, we argue that the residual vector b^r, defined in Definition 6, plays a vital role in the number of selected sets in addition to these cost-efficient rules.

Definition 6 (Residual Vector with respect to a set) Let S_j for $j \in J \setminus \bar{J}$ be an unselected set for a partial solution of the GMOSCP and $b^j \in \{0, 1\}^m$ be the coverage

of the set j. The residual vector $h(j) \in Z^m$ associated with the set S_j is defined as $h(j) = \bar{b} - b^j$, where $\bar{b}$ is the coverage of a partial solution.

Although the cost-efficient sets are the main priority, the number of additional sets needed to satisfy the coverage of items provided by Definition 6 indirectly supports the cost-efficient rules provided in Definition 5. Therefore, we observe a conflict between the cost efficiency of a set and the residual value of an item. We select sets based on lexicographic order preferences which arises, obviously, when such conflicting objectives exist in a decision problem. This optimization ranks preferences by ordering the objective functions according to the importance of the decision maker. For our study, we define the lexicographic orders for the set selection as in Definitions 7 and 8.

Definition 7 *(Lexicographic Order 1 (LO1))* Let $j_1, j_2 \in J \setminus \bar{J}$ be two unselected sets for a partial solution of the GMOSCP and $\lambda \in \mathcal{R}^n_{\geq}$ be a weight vector. We say that the set j_1 is preferred over the set j_2 with respect to the cost-efficient rule g_k, denoted by $j_1 \underset{LO1}{\succeq} j_2$, if $g_k(\lambda, c_{j_1}) \leq g_k(\lambda, c_{j_2})$.

Let J_{best} denote the indices of the cost-efficient sets that can be used to cover the items $e_i \in I \setminus \bar{I}$ for a specified weight vector $\lambda \in \mathcal{R}^p_{\geq}$ according to $LO1$. Then $J_{best}(\lambda)$ is defined as $J_{best}(\lambda) = \arg \min_{S_j : e_i \in S_j} \left\{ g_k(\lambda, c_j) \right\}$ for $k \in K = \{1, 2, 3, 4\}$.

Definition 8 *(Lexicographic Order 2 (LO2))* Let $b^r = b - \bar{b} \in \mathcal{Z}^m$ be a residual vector and $\lambda \in \mathcal{R}^n_{\geq}$ be a weight vector. Let $j_1, j_2 \in J \setminus \bar{J}$ be two unselected sets for a partial solution of the GMOSCP. Let $h(j_1) = b^r - b^{j_1} \in \mathcal{Z}^m$ and $h(j_2) = b^r - b^{j_2} \in \mathcal{Z}^m$ be the two residual vectors associated with the set $j_1, j_2 \in J \setminus \bar{J}$, respectively. We say that the the set j_1 is preferred over the set j_2, denoted by $j_i \underset{LO2}{\succeq} j_2$, if $\sum_{i \in I \setminus \bar{I}} h(j_1) \leq \sum_{i \in I \setminus \bar{I}} h(j_2)$.

We provide the following example to justify the importance of both orders, $LO1$ and $LO2$.

Example 01 Let's consider an instance of a GMOSCP. Let $E = \{e_1, e_2, e_3, e_4\}$ be a set of four items ($m = 4$) and $S_1 = \{e_1, e_2\}, S_2 = \{e_1, e_2, e_4\}, S_3 = \{e_3, e_4\}, S_4 = \{e_1, e_3\}, S_5 = \{e_2\}, S_6 = \{e_1, e_3\}, S_7 = \{e_2, e_4\}$ be seven subsets of E ($n = 7$) with cost vectors $c^1 = (3, 1, 8, 10, 1, 5, 3)$ and $c^2 = (4, 1, 5, 2, 2, 10, 4)$, respectively. Let $b = (2, 1, 2, 1)^T$ be the coverage vector and $\lambda = (0.6, 0.4)^T$ be a weight vector. The initial set density vector is $s^d = (2, 3, 2, 2, 1, 2, 2)^T$. Assume that $\bar{x} = (0, 1, 0, 0, 0, 0, 0)^T$ is a partial solution. The corresponding coverage of $\bar{x}$ is $\bar{b} = (1, 1, 0, 1)^T$. The updated set density and the residual vectors are $s^d = (1, 0, 1, 2, 0, 2, 0)^T$ and $b^r = (1, 0, 2, 0)^T$, respectively. We obtain $\bar{J} = \{2\}, \bar{I} = \{2, 4\}$. Left and right graphs in Fig. 1 illustrates the given GMOSCP and the updated GMOSCP with respect to $\bar{x}$, respectively.

We illustrate the procedure for converting this partial solution $\bar{x}$ by first applying only rule LO1 and then applying both rules $LO1$ and $LO2$. First note that $b^r_2 \leq 0$ and

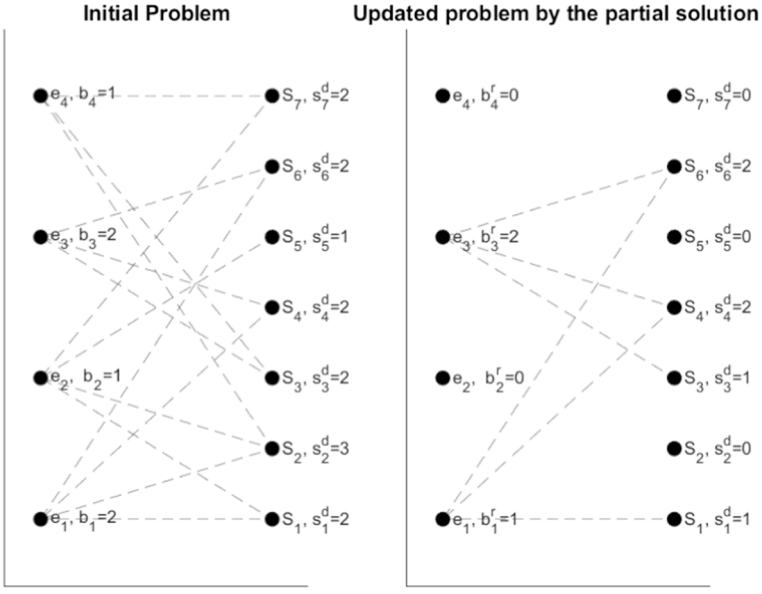

Fig. 1 Graphical Illustration of Example 01

$b_4^r \leq 0$. Thus, we remove the items e_2, e_4 from further consideration. We find $J_{best}(\lambda)$ for the sets with $s_j^d \neq 0$ using the cost-efficient function $g_1(\lambda, c_j)$ defined in Definition 5.

$$J(\lambda) = \arg \min_{S_j : e_i \in S_j} \left\{ \frac{0.6*3+0.4*4}{1}, \frac{0.6*8+0.4*5}{1}, \frac{0.6*10+0.4*2}{2}, \frac{0.6*5+0.4*10}{2} \right\} = \{1, 4\}. \tag{12}$$

Based on Definition 7, the two sets S_1 and S_4 from Eq. (12) are equally likely to be selected.

Case 1: Apply only LO1

1. Assume that we selected the set S_1. We update the density and the residual vectors. Then the partial solution $\bar{x} = (1, 1, 0, 0, 0, 0, 0)^T$ and the coverage of the set S_1 is $b^1 = (1, 0, 0, 0)^T$. The updated residual vector $b^r = (0, 0, 2, 0)^T$. Since $b_1^r \leq 0$ we remove the item e_1 from further consideration, and the updated set density is $s^d = (0, 0, 1, 1, 0, 1, 0)^T$. The only uncovered item at this stage is item e_2, since $b_2^r = 2$, which implies that we need to add two more sets to cover this item.
2. Consider $J(\lambda) = \arg \min_{S_j : e_i \in S_j} \left\{ \frac{0.6*8+0.4*5}{1}, \frac{0.6*10+0.4*2}{1}, \frac{0.6*5+0.4*10}{1} \right\} = \{3, 4\}$. Based on Definition 7, the two sets S_3 and S_4 are equally likely to be selected. Assume that we select the set S_3. We update the density and the residual vectors. Then $\bar{x} = (1, 1, 1, 0, 0, 0, 0)^T$ and the coverage of the set S_3 is $b^3 = (0, 0, 1, 0)^T$. The updated residual and the density vectors are $b^r = (0, 0, 1, 0)^T$ and

$s^d = (0, 0, 0, 1, 0, 1, 0)^T$, respectively. We observe that $b_2^r = 1$, which implies that we need to add one more set to cover this item.

3. Consider $J(\lambda) = \arg \min_{S_j : e_i \in S_j} \left\{ \frac{0.6*10+0.4*2}{1}, \frac{0.6*5+0.4*10}{1} \right\} = \{4\}$. We select S_4 and update the residual vector $b^r = (0, 0, 0, 0)^T$. Since $b_i^r \leq 0$ for all $i \in I$, we have a feasible solution $x = (1, 1, 1, 1, 0, 0, 0)^T$ for the GMOSCP. The corresponding objective vector is $f(x) = (22, 12)^T$.

Case 2: Apply LO1 and LO2

1. The sets S_1 and S_4 from Eq. (12) are equally likely to be selected. Since $|J_{best}(\lambda)| > 1$ with $LO1$, we consider $LO2$. We find $\sum_{i \in I \setminus \bar{I}} h(j_1) = 2$ and $\sum_{i \in I \setminus \bar{I}} h(j_4) = 1$. Since $j_4 \underset{LO2}{\succeq} j_1$, we select the set S_4. The updated residual and the set density vectors are $b^r = (0, 0, 1, 0)^T$, $s^d = (0, 0, 1, 0, 0, 1, 0)^T$, respectively. Since $b_1^r \leq 0$, we remove the item e_1 from further consideration. The only uncovered item at this stage is item e_2, since $b_2^r = 1$, which implies that we need to add one more set to cover this item.
2. Consider $J(\lambda) = \arg \min_{S_j : e_i \in S_j} \left\{ \frac{0.6*8+0.4*5}{1}, \frac{0.6*5+0.4*10}{1} \right\} = \{3\}$. We select the set S_3, and the updated residual vector is $b^r = (0, 0, 0, 0)^T$. Since $b_i^r \leq 0$ for all $i \in I$, we have a feasible solution $x = (0, 1, 1, 1, 0, 0, 0)^T$. The corresponding objective vector is $f(x) = (19, 8)^T$.

This example shows that if we consider only $LO1$, we use three more steps to obtain a feasible solution from $\bar{x}$; if we use both $LO1$ and $LO2$, we use two more steps to obtain a feasible solution from $\bar{x}$. In fact, we observe that the objective vector $(19, 8)^T$ is a nondominated vector comparing to the $\{(19, 8)^T, (22, 12)^T\}$. This example highlights the need for using both $LO1$ and $LO2$ in the algorithm.

Thus $|J_{best}| > 1$ implies that many sets have the same cost-efficient value. Then to identify the best set among the sets with the same cost-efficient values, we select the set that has the highest $h(j)$ value for $j \in J_{best}$ based on $LO2$. If there is a tie with $LO2$, we break it arbitrarily.

3.2 Framework of the algorithm

In this section, we describe each component of the proposed approximation method. The lexicographic set selection approach with multiple cost-efficient rules is integrated into the algorithm proposed in [11]. The new algorithm is called LOLS and presented in Algorithm 1. Starting with the initial population, the algorithm returns an approximation $\hat{Y}$ for the Pareto set Y_p. The algorithm has two significant steps: Initialization (Y_{ini}, X_{ini}) and Obtain Approximation set $\hat{Y}$. The Obtain Approximation step consists of five major procedures: (1) Constructing and solving subproblems (SubProblem); (2) Identifying Partial Solution (FindPartial); (3) Checking Coverage of sets (CheckCoverage); (4) Constructing a Feasible solution (ConvertFeasible); and (5) Improving a Feasible solution (ImproveFeasible).

The LOLS algorithm terminates if it does not find any new nondominated solutions after searching all the feasible regions of the subproblems. Thus, we iterate the algorithm after the initial step until we do not find any new nondominated solutions.

Algorithm 1 LOLS approach

Input: an instance of GMOSCP (I, J, A, b, C), set of weight vectors Λ, initial Population (X_{ini}, Y_{ini})
Step I: Initialization $\hat{X} = \hat{X} \cup X_{ini}$ and $\hat{Y} = \hat{Y} \cup Y_{ini}$
Step II: Obtain Approximation
Set tree height $h = 1$
while termination condition is not fulfilled **do**
    Let $x^h \in \hat{X}$
    $\hat{X} = SubProblem(x^h, \hat{X}, \Lambda, I, J, A, b, C)$
    $h \leftarrow h + 1$
end while
Remove all dominated solutions in $\hat{X}$
Obtain the outcome set $\hat{Y}$ of $\hat{X}$
Return A set of nondominated solutions $\hat{Y}$ for the GMOSCP

3.2.1 Initialization

In the initialization procedure, we construct and solve the achievement scalarization optimization problems given in (11) for the GMOSCP by using some user-defined preference weight vectors. Our preliminary work demonstrated that achievement scalarization optimization problems provided more Pareto points than the weighted-sum optimization problems for the same number of weight vectors. Further, the Pareto points are almost evenly distributed within the Pareto set with this method.

We define the number of weight vectors as a parameter. To obtain a reasonable approximation of the Pareto set, we need to solve a number of scalarized problems [36]. Proposition 2.1 implies that these solutions are efficient solutions of the GMOSCP. Here, we take the ideal point f^{ideal} as the reference point. Thus, $f_q^{ideal} = \min_{x^* \in X} f_q(x^*)$. We use these efficient solutions as initial solutions X_{ini}. We evaluate the objective function f_q for $q \in Q$ at these initial solutions to find the corresponding initial values for Y_{ini}.

3.2.2 Construct and solve subproblems

The technique for constructing and solving subproblems of the GMOSCP is given in Procedure 2. This is a divide-and-conquer approach. The procedure divides the GMOSCP into subproblems similar to the original problem and recursively solves the subproblems. The procedure then combines the solutions of the subproblems to solve the GMOSCP. Because divide-and-conquer techniques solves subproblems recursively, each subproblem must be more straightforward than the original problem. There must be a systematic approach to construct subproblems. We now give an explanation of the procedure.

For each solution $x^h \in \hat{X}$, the branching constraints, denoted by $N(x^h, x) \leq L$ and $N(x^h, x) \geq L + 1$, are constructed where $N(x^h, x) \leq L$ denotes the neighboring solutions of x^h within L distance. Here the distance L is the Hamming distance

[17], which is a positive integer. When $h = 1$, the feasible region of subproblem p^h is defined as $X \cap \{N(x^h, x) \leq L\}$ and this subregion is searched for new nondominated points. We continue to partition region $X \cap \{N(x^h, x) \geq L + 1\}$ by adding the branching constraints defined with respect to new nondominated solutions. For $x^{h+1} \in \hat{X}$ the algorithm considers previously unexplored subregion and partitions it by adding the branching constraint $N(x^{h+1}, x) \leq L$ to define the subproblem p^{h+1}. The algorithm keeps partitioning the feasible region X until no nondominated solutions are found. The last subregion is searched separately since no more new nondominated solutions are available to continue the partitioning. Each search space is extensively searched for new solutions by performing several searches and the partitioning approach can be explained using a tree structure [11]. To identify the nondominated points in each subregion, we convert the vector objective function to a scalarized objective function $f_{ws}(\lambda)$ using weighted-sum scalarization described in Sect. 2.2.1 and also replace the integer variables with nonnegative variables (relaxed variables). We solve these modified subproblems. Here, for each subproblem, we solve the relaxed problems defined in model 6 for $\lambda \in \Lambda$. The number of relaxed sub-problems defined for each subproblem will be equal to the number of vectors in Λ. Then we send the optimal solutions of the subproblems to Procedure 3 to identify the partial solution for the GMOSCP. A set of Λ weight vectors can be used in this procedure for the scalarization. However, the size of the set Λ is not clear in advance. Therefore, we consider it also as a numerical parameter as described and analyzed in Sect. 5.

Procedure 2 Construct and solve subproblem (SubProblem)

Input: x^h: a solution to GMOSCP and $\hat{X}, \Lambda, I, J, A, b, C$
Initialization: Set $\bar{X} = \emptyset$
for $\lambda \in \Lambda$ **do** ▷ exploration of subproblems
  Construct the subproblem p^h utilizing λ
  $x \leftarrow$ Solution of the subproblem p^h ▷ A linear solver can be used here
  Partial solution $(\bar{x}, b^r, \bar{I}, \bar{J}) \leftarrow FindPartial(x, I, J, A, b, C)$
  Feasible solution $\hat{x} \leftarrow ConvertFeasible(\bar{x}, b^r, \bar{I}, \bar{J}, I, J, A, b, C)$
  Improved solution $\hat{x} \leftarrow ImproveFeasible(\bar{x}, b^r, I, J, A, b, C)$
  if $f(\hat{x})$ is nondominated **then**
    Update $\hat{X} = \bar{X} \cup \hat{x}$ and remove all the solutions dominated by $\hat{x}$
  else Discard $\hat{x}$
  end if
end for
Return: $\hat{X}$

3.2.3 Identify the partial solutions

We present the approach for identifying a partial solution in Procedure 3. A partial solution is constructed by identifying desirable sets and updating coverage requirements for items based on the desirable sets using an optimal solution $\bar{x}$ of subproblem p^h. Here, we assume that $\bar{x}_j$ is a desirable candidate if $x_j \geq 0.5$. If $\bar{x}_j$ is a desirable candidate for a feasible solution, we make $\bar{x}_j = 1$ and add the index j to $\bar{J}$, where $\bar{J}$ is the index set of the selected sets to construct a feasible solution. We

also identify the items covered by the set S_j using Procedure 4 and update the corresponding residual values.

Procedure 3 Identify partial solution (FindPartial)

1: **Input:** x: optimal solution of a subproblem, I, J, A, b, C
2: **Initialization:** Set $\bar{x} \in \{0\}^n, b^r = b, \bar{I} = \emptyset, \bar{J} = \emptyset$
3: **for** each x_j value, $j \in J$ **do**
4: **if** x_j is desirable **then**
5: Update $\bar{J} = \bar{J} \cup \{j\}$ and set $\bar{x}_j = 1$
6: $(b^r, \bar{I}) \leftarrow CheckCoverage(I, \bar{I}, S_j, b^r)$
7: **end if**
8: **end for**
9: *Return:* $\bar{x}, b^r, \bar{I}, \bar{J}$

3.2.4 Identify coverage

Procedure 4 is used to identify the covered items by a given set and the corresponding residual vector. If the residual value for a given item e_i is nonpositive, that means the coverage requirement of the item has been satisfied. Thus, the procedure adds the corresponding index to the index set of the covered items $\bar{I}$ and set $b_i^r = 0$. We also update the set densities in this procedure.

3.2.5 Construct feasible solutions

We present the approach for constructing a feasible solution in Procedure 5. The partial solution $\bar{x}$ produced by the subproblem p^h is analyzed by this procedure. For the unselected sets S_j, for $j \in J \setminus \bar{J}$, the cost-efficient sets are identified using Definition 7 according to a cost-efficient rule $g_k(\lambda, c_j)$. If $|J_{best}(\lambda) > 1|$, then Definition 8 will be used to break the ties. Note that if more than one candidate set is available according to Definition 8, we break the ties arbitrarily and one of the four cost-efficient rules given in Definition 5 will be used. After identifying the best set, the covered items and the updated residual vector are obtained using the *CheckCoverage* procedure. The algorithm will continue the procedure until all items are covered and it returns a feasible solution based on cost-efficient rule $g_k(\lambda, c_j)$.

Procedure 4 Identify Coverage (CheckCoverage)

Input: $I, \bar{I}, S_j, b^r$
for items $e_i,\ i \in I \setminus \bar{I}$ **do**
  if item e_i can be covered by set S_j **then**
    update the residual $b_i^r = b_i^r - 1$
    if $b_i^r \leq 0$ **then**
      update $\bar{I} = \bar{I} \cup \{i\}$
      Set $b_i^r = 0$
      set $a_{ij} = 0$
      update set density for all $j \in J \setminus \bar{J}$
    end if
  end if
end for
Return: $b^r, \bar{I}$

3.2.6 Improve feasible solutions

We present the approach for improving a feasible solution in Procedure 6. The feasible solution $\bar{x}$ produced by the subproblem p^h is improved by this procedure. For each set $j \in J$ such that $\bar{x}_j = 1$, we check whether we obtain a feasible solution after removing the set S_j. All such sets are identified. We define a scalar cost for each such set S_j as $\sum_{q \in Q} c_j^q$. Then we set $\bar{x}_j = 0$ for the solution $\bar{x}$ starting from the highest scalar cost of such sets while maintaining the feasibility of the solution. We note that different preference rules can be used to improve the solutions at this stage.

Procedure 5 Converting to a feasible solution (ConvertFeasible)

Input: $\bar{x}, b^r, \bar{I}, \bar{J}, \lambda, I, J, b, C$
while $\bar{I} \neq I$ **do** Find $|J_{best}(\lambda)|$ using Definition 7
  if $|J_{best}(\lambda) = 1|$ **then**
    Select the set S_{j^*} for $j^* \in J_{best}(\lambda)$
  end if
  if $|J_{best}(\lambda) > 1|$ **then**
    Identify the set S_{j^*} using Definition 8
  end if
  Update $\bar{J} = \bar{J} \cup \{j^*\}$ and set $\bar{x}_{j^*} = 1$
  $\bar{I} \leftarrow CheckCoverage(I, \bar{I}, S_{j^*}, b^r)$
  Update the set densities S_j^q for $J \setminus \bar{J}$
end while

Return: $\bar{x}$

According to this algorithm, the maximum number of iterations needed to convert an infeasible solution to a feasible solution is $\sum_{i \in I} b_i - |x|$.

4 Experimental outline

In this section, we present our test problems and performance metrics used to verify the algorithm's performance. We also provide our approach to finding weight vectors for scalarizing the objective vector. We used a personal laptop equipped with an Intel I-7 processor and 8 GB memory for implementation.We implement the algorithm using MATLAB 2020 version. We use MATLAB intlinprog function in the initial stage to solve the scalarized optimization problems and lingprog function to solve the relaxed-scalarized optimization problems. Finally, we utilize MATLAB random number generator 'rng(6)', where 6 specifies the seed with 'twister', to generate random test problems.

4.1 Test problems

In this study, diverse test instances of the GMOSCP are considered as the test problems. We consider two sources for data: test problems from literature and randomly generated test problems. We conducted multiple analyses.

Test problems collected from [19] are called *scp*(l, m, n), where l, m, and n stand for the test group, the number of items, and the number of sets in the test instances, respectively. Test problems provided in [19] are designed for the MOSCP with two objectives. Since our algorithm can be applied to the MOSCP, we use it without modification for comparison with the state-of-the-art MOSCP algorithms. For instance, *scp*(A, 40, 200) means the test instance is from group A and has 40 items and 200 sets. We expanded this data for the GMOSCP. First, we consider $b_i \in [b_l, b_u]$, where $b_l, b_u \in Z_+$ are upper bound and lower bound values for $b_i \; \forall \, i \in I$, respectively, and we identified the total coverage for each item ($\sum_{j \in J} a_{ij}, \; \forall \, i \in I$). If $\sum_{j \in J} a_{ij} \leq b_i$ for some $i \in I$, then we set $b_i = 1$. This modification guarantees the feasibility of the expanded test problems. These problems are denoted by *gscp*(l, m, n), where l, m, and n stand for the test group, the number of items, and the number of sets in the test instances, respectively. Second, we generated random test problems using MATLAB random number generator. We specify m and n. Then we generated a random number a_{ij}, $\forall \, i \in I$, $j \in J$. If $a_{ij} \geq 0.5$, we set $a_{ij} = 1$, otherwise we set $a_{ij} = 0$. If $\sum_{j \in J} a_{ij} = 0$ for some $i \in I$, then we set $a_{ij} = 1$ for some $j \in J$. We used $b_i \in [b_l, b_u] \; \forall \, i \in I$, and, as before, if $\sum_{j \in J} a_{ij} < b_i$ for some $i \in I$, then we set $b_i = 1$. This procedure generates feasible GMOSCP test instances. The cost coefficients are generated as $c_j^q \in [c_l, c_u]$, where $c_l, c_u \in Z_+$ are the lower bound and upper bound values, respectively, for c_j^q for $q \in Q$, $j \in J$. These test problems are called *rscp*(m, n). For instance, *rscp*(30, 500) means the random test instance with 30 items and 500 sets.

4.2 Performance metrics

We used two metrics that are the most frequently used measures in multi-objective optimization for analyzing the the performance of the LOLS algorithm.

Table 1 $\mathcal{C}$-metric values of the approximations obtained by LOLS for $L = 7$

Instances	$\hat{Y}_{g_1}$ vs$\hat{Y}_{g_2}$		$\hat{Y}_{g_1}$ vs $\hat{Y}_{g_3}$		$\hat{Y}_{g_1}$ vs $\hat{Y}_{g_4}$		$\hat{Y}_{g_2}$ vs $\hat{Y}_{g_3}$		$\hat{Y}_{g_2}$ vs $\hat{Y}_{g_4}$		$\hat{Y}_{g_3}$ vs $\hat{Y}_{g_4}$	
	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$
$b_i \in [1,3],\ \forall\, i \in I$												
gscp(A,10,100)	**0.6571**	0.6667	**0.6571**	0.6667	**0.6190**	0.8056	1.0000	1.0000	**0.6190**	0.7143	**0.6190**	0.7143
gscp(B,10,100)	1.0000	**0.4333**	1.0000	**0.4333**	**0.7941**	1.0000	1.0000	1.0000	**0.3824**	1.0000	**0.3824**	1.0000
gscp(C,10,100)	1.0000	1.0000	1.0000	1.0000	**0.9091**	1.0000	1.0000	1.0000	**0.9091**	1.0000	**0.9031**	1.0000
gscp(D,10,100)	1.0000	**0.9333**	1.0000	**0.9333**	1.0000	1.0000	1.0000	1.0000	**0.9333**	1.0000	**0.9333**	1.0000
gscp(A,40,200)	0.9412	**0.5385**	0.9412	**0.5385**	**0.6176**	0.9615	1.0000	1.0000	**0.4118**	1.0000	**0.4118**	1.0000
gscp(B,40,200)	0.8000	**0.5000**	0.8000	**0.5000**	**0.7500**	0.8125	1.0000	1.0000	**0.5000**	0.8000	**0.5000**	0.8000
gscp(C,40,200)	1.0000	**0.9167**	1.0000	**0.9167**	1.0000	1.0000	1.0000	1.0000	**0.9167**	1.0000	**0.9167**	1.0000
gscp(D,40,200)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
gscp(A,40,400)	1.0000	**0.5500**	1.0000	**0.5500**	**0.4054**	0.8500	1.0000	1.0000	**0.2973**	1.0000	**0.2973**	1.0000
gscp(B,40,400)	0.9444	**0.2821**	0.9444	**0.2821**	**0.2836**	0.7949	1.0000	1.0000	**0.1642**	1.0000	**0.1642**	1.0000
gscp(C,40,400)	1.0000	**0.5000**	1.0000	**0.5000**	0.9091	**0.7273**	1.0000	1.0000	**0.5000**	1.0000	**0.5000**	1.0000
gscp(D,40,400)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
gscp(A,60,600)	1.0000	**0.3929**	1.0000	**0.3929**	**0.2391**	1.0000	1.0000	1.0000	**0.2391**	1.0000	**0.2391**	1.0000
gscp(B,60,600)	1.0000	**0.4800**	1.0000	**0.4800**	**0.1846**	0.8800	1.0000	1.0000	**0.1692**	1.0000	**0.1692**	1.0000
gscp(C,60,600)	1.0000	**0.9091**	1.0000	**0.9091**	1.0000	1.0000	1.0000	1.0000	**0.9091**	1.0000	**0.9091**	1.0000
gscp(D,60,600)	1.0000	**0.9167**	1.0000	**0.9167**	1.0000	1.0000	1.0000	1.0000	**0.9167**	1.0000	**0.9167**	1.0000
gscp(A,80,800)	1.0000	**0.5789**	1.0000	**0.5789**	**0.3235**	1.0000	1.0000	1.0000	**0.3235**	1.0000	**0.3235**	1.0000
gscp(B,80,800)	1.0000	**0.7333**	1.0000	**0.7333**	**0.2619**	1.0000	1.0000	1.0000	**0.2619**	1.0000	**0.2619**	1.0000
gscp(C,80,800)	1.0000	**0.7692**	1.0000	**0.7692**	**0.8462**	1.0000	1.0000	1.0000	**0.7692**	1.0000	**0.7692**	1.0000
gscp(D,80,800)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
gscp(A,100,1000)	0.8947	**0.5455**	0.8947	**0.5455**	**0.5200**	0.8636	1.0000	1.0000	**0.4800**	0.8947	**0.4800**	0.8947
gscp(B,100,1000)	0.9412	**0.4583**	0.9412	**0.4583**	**0.6250**	0.6667	1.0000	1.0000	**0.4583**	1.0000	**0.4583**	1.0000
rscp(20,500)	0.8696	**0.7500**	0.8696	**0.7500**	0.9565	**0.8333**	1.0000	1.0000	0.8261	0.8261	0.8261	0.8261

Definition 9 C-Metrics [51] Let $R, S \subset X$ be two solution sets of GMOSCP. The metric $\mathcal{C}(R, S)$ specifies the fraction of the solutions in the set S that are dominated by at least one solution in the set R

$$\mathcal{C}(R, S) = \frac{|\{a'' \in S : \exists\, a' \in R : a' \text{ dominates } a''\}|}{|S|}$$

The value $\mathcal{C}(R, S) = 1$ implies that all solutions in S are dominated by or equal to solutions in R, while the $\mathcal{C}(R, S) = 0$ indicates that none of the solutions in S is dominated by solutions in R. It is important to note that when comparing different approximations' nondominated sets, both $\mathcal{C}(R, S)$ and $\mathcal{C}(S, R)$ have to be considered, since $\mathcal{C}(R, S)$ is not necessarily equal to $\mathcal{C}(S, R)$.

Definition 10 Hypervolume measure [51] The $\mathcal{H}$ measure gives the volume of the portion of the objective space that is dominated by the Pareto set from below and bounded by the nadir point from above. For two sets R and S, the set R is better than the set S with respect to the hypervolume measure if $\mathcal{H}(R) > \mathcal{H}(S)$.

4.3 Number of scalarization to obtain the initial population

Several strategies can be used to find weight vectors in Λ for the initial stage and constructive stage of the subproblem. We use the uniformly distributed normalized vectors proposed in [20, 36]. For a weight vector $\lambda \in \Lambda$, each individual component λ_q takes one of the values given in the set $\{l/s,\ l = 0, 1, \ldots, s\}$ such that $\sum_{q \in Q} \lambda_i = 1$.

In our study we set $s = 10$. When $p = 2$, each component $[\lambda_1, \lambda_2]$ of $\lambda \in \Lambda$ can be written as $[l/s, 1 - l/s]$, where $l = 0, 1, \ldots, 10$. Thus we solve 11 scalarized problems using ASF in the initial stage. This guarantees that $\sum_{q \in Q} \lambda_i = 1$.

5 Experimental results

In this section, we present the numerical experiments that we conducted to test the LOLS algorithm's performance in various test instances. In our computational experiments, we analyzed two main components of the LOLS algorithm. Both components are important for the approximation quality. The first component is the neighborhood length denoted by L, and the second component is the effects of the four different cost-efficient rules. We experimented with several values of L and obtained better results for $L = 5, 6, 7$. We did not observe significant improvements for smaller or larger values of L. Therefore, in the following sections, we have included numerical results only for $L = 5$ or $L = 7$ or both. We also compare the approximations produced by each cost-efficient rule $g_k(\lambda, x)$ for $k = 1, 2, 3, 4$. We denote the approximation sets produced by the four different cost-efficient rules by $\hat{Y}_{g_k}$ for $k = 1, 2, 3, 4$. We define $\hat{Y}$, the approximation set produced by LOLS, as the nondominated set of $\cup_{k=1}^{4} \hat{Y}_{g_k}$. We provide the values of all the parameters in the corresponding tables.

5.1 Comparisons of priority rules based on $\mathcal{C}$ metric

In this section, each cost-efficient rule's effect is compared (pair-wise) regarding the $\mathcal{C}$ metric. The results are presented in Table 1 for $L = 7$. The set R corresponds to the rule g_{k_1} and the set S corresponds to the rule g_{k_2} for $k_1 \leq k_2$. Here, the smaller the metric, the better the approximation. The best-performing rule appears in bold. In the case of a tie, neither is bolded. We observe a significant performance difference among the rules when comparing the test instances using the $\mathcal{C}$ metric. It can be observed that the cost-efficient rule g_4 outperforms the others on almost all test instances, although g_1 performs slightly better than g_4 on some instances. The rules g_2 and g_3 provide the same metric. We observe almost the same performance rate with $L = 5$.

5.2 Effect of search length

In this section, the effect of search length on approximations produced by each cost-efficient rule is compared with respect to the $\mathcal{H}$ measures. The results are presented in Table 2 for $L = 5$ and $L = 7$. We note that improving the search length did not improve the results significantly for the test problems used in this study. Here, the higher the metric, the better the approximation. The best-performing rule appears in bold. Confirming the observations we made on the $\mathcal{C}$ metric, it can be observed that the cost-efficient rule g_4 outperforms the others on most test instances, although rule g_1 performs slightly better than rule g_4 on a few test instances. This observation confirms that in contrast to cost-efficient rules g_1, g_2, g_3, the rule g_4 enable us to enables us to provide more priority for the cost-efficient value. We did not observe a significant difference when $\mathcal{H}$ measure was applied to the combined approximation $\hat{Y}$ with $L = 5$ and $L = 7$.

Table 3 summarizes the results of the $\mathcal{H}$ measure. For example, out of 32 $gscp(l, p, m, n)$ test instances, 26 test instances provide a higher $\mathcal{H}$ measure for rule g_4 when $L = 5$ (approximately 81.82%). Overall 81.48% and 88.89% of the total test problems, the rule g_4 performs well when $L = 5$ and $L = 7$, respectively. This analysis also confirms the observation we made based on the $\mathcal{C}$ metric. Overall, the rule g_4 outperforms the other rules.

5.3 Computational time of the LOLS

The computational time of the algorithm varies with the size of the test problem and the vector b. Table 4 shows the mean computational times. We group the test instances based on the number of sets. As we had expected, the run time increases as L increases. We observed a significantly high computational time for the test instances with more than 600 sets when $L = 7$ compared to when $L = 5$. Since we do not observe a dramatic change of the measures with increasing the value of L, our study concludes that a small search length like $L = 5$ is enough, though it is an experimental factor.

Table 1 (continued)

Instances	$\hat{Y}_{g_1}$ vs$\hat{Y}_{g_2}$		$\hat{Y}_{g_1}$ vs $\hat{Y}_{g_3}$		$\hat{Y}_{g_1}$ vs $\hat{Y}_{g_4}$		$\hat{Y}_{g_2}$ vs $\hat{Y}_{g_3}$		$\hat{Y}_{g_2}$ vs $\hat{Y}_{g_4}$		$\hat{Y}_{g_3}$ vs $\hat{Y}_{g_4}$	
	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$
rscp(75,600)	0.7812	**0.7353**	0.7812	**0.7353**	0.9118	**0.8529**	1.0000	1.0000	**0.7353**	0.7500	**0.7353**	0.7500
rscp(20,700)	0.9630	0.9630	0.9630	0.9630	**0.9310**	0.9630	1.0000	1.0000	**0.8966**	0.9259	**0.8966**	0.9259
rscp(100,750)	0.8387	**0.7879**	0.8387	**0.7879**	0.8750	**0.8485**	1.0000	1.0000	0.8125	**0.8065**	0.8125	**0.8065**
rscp(70,900)	0.8529	**0.6750**	0.8529	**0.6750**	**0.8000**	0.9000	1.0000	1.0000	**0.5750**	0.8235	**0.5750**	0.8235
rscp(100,1000)	0.8182	**0.7619**	0.8182	**0.7619**	**0.6818**	0.9048	1.0000	1.0000	**0.6364**	0.7576	**0.6364**	0.7576
rscp(250,1000)	0.7083	**0.6667**	0.7083	**0.6667**	0.8605	**0.5667**	1.0000	1.0000	**0.6047**	0.6875	**0.6047**	0.6875
rscp(100,1500)	0.8529	**0.8710**	**0.8529**	0.8710	**0.8125**	0.9355	1.0000	1.0000	**0.7500**	0.8824	**0.7500**	0.8824
rscp(200,1500)	0.8654	**0.8600**	0.8654	**0.8600**	**0.8679**	0.9400	1.0000	1.0000	**0.7925**	0.8654	**0.7925**	0.8654
rscp(200,2000)	0.7609	**0.6875**	0.7609	**0.6875**	0.8444	**0.7292**	1.0000	1.0000	0.6889	**0.6739**	0.6889	**0.6739**
$b_i = 1,\ \forall\, i \in I$												
scp(A,10,100)	1.0000	1.0000	1.0000	1.0000	**0.8710**	1.0000	1.0000	1.0000	**0.8710**	1.0000	**0.8710**	1.0000
scp(B,10,100)	1.0000	**0.9118**	1.0000	**0.9118**	**0.8788**	0.9118	1.0000	1.0000	**0.8788**	0.9375	**0.8799**	0.9375
scp(C,10,100)	1.0000	**0.8889**	1.0000	**0.8889**	**0.8889**	1.0000	1.0000	1.0000	**0.7778**	1.0000	**0.7778**	1.0000
scp(D,10,100)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
scp(A,40,200)	1.0000	**0.9890**	1.0000	**0.989**	0.9667	**0.9451**	1.0000	1.0000	0.9556	**0.9444**	0.9556	**0.9444**
scp(B,40,200)	1.0000	**0.9794**	1.0000	**0.9794**	**0.9505**	0.9794	1.0000	1.0000	**0.9406**	0.9896	**0.9406**	0.9896
scp(C,40,200)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
scp(D,40,200)	1.0000	**0.9167**	1.0000	**0.9167**	1.0000	**0.9722**	1.0000	1.0000	**0.9143**	0.9697	**0.9143**	0.9697
scp(A,40,400)	0.9829	**0.9492**	0.9829	**0.9492**	0.9774	0.9774	1.0000	1.0000	**0.9435**	0.9771	**0.9435**	0.9771
scp(B,40,400)	0.9655	**0.8710**	0.9655	**0.8710**	**0.9000**	0.9570	1.0000	1.0000	**0.8263**	0.9598	**0.8263**	0.9598
scp(C,40,400)	1.0000	**0.9286**	1.0000	**0.9286**	**0.8636**	0.9286	1.0000	1.0000	**0.7955**	0.9231	**0.7955**	0.9231
scp(D,40,400)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
scp(A,60,600)	0.9764	**0.7718**	0.9764	**0.7718**	**0.7267**	0.9396	1.0000	1.0000	**0.6163**	0.9528	**0.6163**	0.9528

Table 1 (continued)

Instances	$\hat{Y}_{g_1}$ vs$\hat{Y}_{g_2}$		$\hat{Y}_{g_1}$ vs $\hat{Y}_{g_3}$		$\hat{Y}_{g_1}$ vs $\hat{Y}_{g_4}$		$\hat{Y}_{g_2}$ vs $\hat{Y}_{g_3}$		$\hat{Y}_{g_2}$ vs $\hat{Y}_{g_4}$		$\hat{Y}_{g_3}$ vs $\hat{Y}_{g_4}$	
	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$	$\mathcal{C}(R,S)$	$\mathcal{C}(S,R)$
scp(B,60,600)	0.9286	**0.7513**	0.9286	**0.7513**	**0.7699**	0.9067	1.0000	1.0000	**0.6460**	0.9011	**0.6460**	0.9011
scp(C,60,600)	1.0000	**0.9375**	1.0000	**0.9375**	**0.7619**	1.0000	1.0000	1.0000	**0.7143**	1.0000	**0.7143**	1.0000
scp(D,60,600)	0.9744	**0.9722**	0.9744	**0.9722**	**0.7955**	0.9722	1.0000	1.0000	**0.7727**	0.9744	**0.7727**	0.9744

Bold indicates the best measure

Table 2 Comparison of search length L in terms of $\mathcal{H}$ values for various GMOSCP instances

Instances	$L=5$					$L=7$					
	$\mathcal{H}(\hat{Y}_{g_1})$	$\mathcal{H}(\hat{Y}_{g_2})$	$\mathcal{H}(\hat{Y}_{g_3})$	$\mathcal{H}(\hat{Y}_{g_4})$	$\mathcal{H}(\hat{Y})$	$\mathcal{H}(\hat{Y}_{g_1})$	$\mathcal{H}(\hat{Y}_{g_2})$	$\mathcal{H}(\hat{Y}_{g_3})$	$\mathcal{H}(\hat{Y}_{g_4})$	$\mathcal{H}(\hat{Y})$	$\mathcal{H}(P)$
$b_i \in [1,3],\ \forall\, i \in I$											
gscp(A,10,100)	**0.7564**	0.7512	0.7512	0.7551	**0.7592**	0.7524	0.7572	0.7572	**0.7561**	**0.7609**	0.7682
gscp(B,10,100)	0.7733	0.763	0.763	**0.7761**	**0.7764**	0.7742	0.7515	0.7515	**0.7773**	**0.7773**	0.7846
gscp(C,10,100)	**0.6541**	0.6488	0.6488	0.6488	**0.6548**	0.6541	0.6541	0.6541	**0.6545**	**0.6545**	0.6604
gscp(D,10,100)	**0.5623**	0.5621	0.5621	**0.5623**	**0.5623**	**0.5623**	0.5621	0.5621	**0.5623**	**0.5623**	0.5758
gscp(A,40,200)	0.7368	0.7295	0.7295	**0.7403**	**0.7408**	0.7346	0.7244	0.7244	**0.7394**	**0.7394**	0.7646
gscp(B,40,200)	0.797	0.7839	0.7839	**0.8026**	**0.8031**	0.795	0.7839	0.7839	**0.7994**	**0.8010**	0.8268
gscp(C,40,200)	**0.7591**	0.7549	0.7549	**0.7591**	**0.7591**	0.7591	0.7549	0.7549	**0.7591**	**0.7591**	0.8144
gscp(D,40,200)	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**	**0.8058**	0.8443
gscp(A,40,400)	0.8012	0.7842	0.7842	**0.805**	**0.8053**	0.7999	0.7842	0.7842	**0.8041**	**0.8044**	0.8313
gscp(B,40,400)	0.838	0.8197	0.8197	**0.8456**	**0.8460**	0.8369	0.8201	0.8201	**0.8446**	**0.8456**	0.8640
gscp(C,40,400)	0.7257	0.7224	0.7224	**0.7273**	**0.7276**	**0.7282**	0.7184	0.7184	0.7267	**0.7282**	0.7575
gscp(D,40,400)	**0.8239**	**0.8239**	**0.8239**	**0.8239**	**0.8239**	**0.8239**	**0.8239**	**0.8239**	**0.8239**	**0.8239**	0.8444
gscp(A,60,600)	0.8433	**0.8735**	**0.8735**	0.8548	**0.8548**	0.8434	0.8353	0.8353	**0.8534**	**0.8534**	0.8800
gscp(B,60,600)	0.8204	0.8149	0.8149	**0.8326**	**0.8326**	0.8203	0.814	0.814	**0.8318**	**0.8318**	0.8584
gscp(C,60,600)	0.8482	0.8481	0.8481	**0.8482**	**0.8482**	**0.8482**	0.8481	0.8481	**0.8482**	**0.8482**	0.8865
gscp(D,60,600)	**0.8415**	0.8414	0.8414	**0.8415**	**0.8415**	0.8415	0.8414	0.8414	**0.8415**	**0.8415**	0.8876
gscp(A,80,800)	0.7799	0.777	0.777	**0.7902**	**0.7902**	0.7798	0.777	0.777	**0.7894**	**0.7894**	0.8302
gscp(B,80,800)	0.8087	0.8042	0.8042	**0.8234**	**0.8234**	0.8088	0.8042	0.8042	**0.8234**	**0.8234**	0.8549
gscp(C,80,800)	0.9146	0.8859	0.8859	**0.9155**	**0.9155**	0.9146	0.8859	0.8859	**0.9148**	**0.9148**	0.9408
gscp(D,80,800)	**0.9361**	**0.9361**	**0.9361**	**0.9361**	**0.9361**	**0.9361**	**0.9361**	**0.9361**	**0.9361**	**0.9361**	0.9706
gscp(A,100,1000)	0.8774	0.8625	0.8703	**0.8806**	**0.8811**	0.8774	0.8625	0.8703	**0.8806**	**0.8811**	0.8825
gscp(B,100,1000)	0.8549	0.8508	0.8508	**0.8569**	**0.8580**	0.8549	0.8508	0.8508	**0.8569**	**0.8580**	0.8680
rscp(20,500)	0.8883	0.8879	0.8879	**0.8867**	**0.8890**	**0.8870**	0.8853	0.8853	0.8849	**0.8878**	0.8895

Table 2 (continued)

Instances	$L = 5$					$L = 7$					
	$\mathcal{H}(\hat{Y}_{g_1})$	$\mathcal{H}(\hat{Y}_{g_2})$	$\mathcal{H}(\hat{Y}_{g_3})$	$\mathcal{H}(\hat{Y}_{g_4})$	$\mathcal{H}(\hat{Y})$	$\mathcal{H}(\hat{Y}_{g_1})$	$\mathcal{H}(\hat{Y}_{g_2})$	$\mathcal{H}(\hat{Y}_{g_3})$	$\mathcal{H}(\hat{Y}_{g_4})$	$\mathcal{H}(\hat{Y})$	$\mathcal{H}(P)$
rscp(75,600)	0.9198	0.9201	0.9202	**0.9216**	**0.9224**	**0.9227**	0.9222	0.9222	0.9226	**0.9235**	0.9265
rscp(20,700)	0.9555	0.9545	0.9545	**0.9555**	**0.9557**	0.9524	0.9524	0.9524	**0.9526**	**0.9528**	0.9569
rscp(100,750)	0.9219	0.9249	0.9249	**0.9228**	**0.925**	**0.9241**	0.9233	0.9233	0.9229	**0.9248**	0.9329
rscp(70,900)	0.9510	0.9500	0.9500	**0.9513**	**0.9517**	0.9492	0.9494	0.9494	**0.9512**	**0.9514**	0.9549
rscp(100,1000)	**0.9506**	0.9486	0.9486	0.9506	**0.9508**	0.9502	0.9493	0.9493	**0.9511**	**0.9518**	0.9560
rscp(250,1000)	0.9407	0.9397	0.9397	0.9391	**0.9418**	0.9401	0.9394	0.9394	**0.9402**	**0.9410**	0.9476
rscp(100,1500)	0.9581	0.9579	0.9579	**0.9582**	**0.9589**	0.9575	0.9557	0.9557	**0.9574**	**0.9579**	0.9638
rscp(200,1500)	**0.9569**	0.9568	0.9568	0.9562	**0.9573**	0.9567	0.9564	0.9564	**0.9568**	**0.9635**	0.9635
rscp(200,2000)	**0.9641**	0.9637	0.9637	0.9620	**0.9645**	**0.9655**	0.9644	0.9644	0.9644	**0.9649**	0.9698
$b_i = 1,\ \forall\, i \in I$											
scp(A,10,100)	**0.8156**	**0.8156**	**0.8156**	**0.8156**	**0.8156**	0.8149	0.8149	0.8149	**0.8160**	**0.8160**	0.8170
scp(B,10,100)	0.7218	0.7218	0.7218	**0.7220**	**0.7220**	**0.7202**	0.7190	0.7190	**0.7202**	**0.7216**	0.7241
scp(C,10,100)	0.5477	0.5477	0.5477	**0.5488**	**0.5458**	0.5485	0.5298	0.5298	**0.5488**	**0.5488**	0.5496
scp(D,10,100)	0.4709	0.4709	0.4709	**0.4709**	**0.4709**	**0.4710**	**0.4710**	**0.4710**	**0.4710**	**0.4710**	0.4709
scp(A,40,200)	**0.7769**	0.7769	0.7769	0.7767	**0.7769**	**0.7769**	**0.7769**	**0.7769**	0.7767	**0.7769**	0.7774
scp(B,40,200)	**0.8631**	0.8629	0.8629	**0.8631**	**0.8632**	**0.8631**	0.8629	0.8629	**0.8631**	**0.8631**	0.8637
scp(C,40,200)	**0.7187**	**0.7187**	**0.7187**	**0.7187**	**0.7187**	**0.7187**	**0.7187**	**0.7187**	**0.7187**	**0.7187**	0.7344
scp(D,40,200)	**0.8831**	0.8824	0.8824	**0.8831**	**0.8831**	**0.8831**	0.8824	0.8824	**0.8831**	**0.8831**	0.8911
scp(A,40,400)	0.8435	0.8434	0.8434	**0.8437**	**0.8437**	0.8435	0.8434	0.8434	**0.8437**	**0.8437**	0.8439
scp(B,40,400)	0.8788	0.8786	0.8786	**0.8790**	**0.8790**	**0.8788**	0.8786	0.8786	0.8790	**0.8790**	0.8794
scp(C,40,400)	0.7576	0.7571	0.7571	**0.7581**	**0.7583**	0.7576	0.7571	0.7571	**0.7581**	**0.7583**	0.7645
scp(D,40,400)	**0.6753**	**0.6753**	**0.6753**	**0.6753**	**0.6753**	**0.6753**	**0.6753**	**0.6753**	**0.6753**	**0.6753**	0.6885
scp(A,60,600)	**0.8785**	0.8776	0.8500	0.8508	**0.8790**	**0.8785**	0.8776	0.8776	0.8790	**0.8790**	0.8797

Table 2 (continued)

Instances	$L = 5$					$L = 7$					
	$\mathcal{H}(\hat{Y}_{g_1})$	$\mathcal{H}(\hat{Y}_{g_2})$	$\mathcal{H}(\hat{Y}_{g_3})$	$\mathcal{H}(\hat{Y}_{g_4})$	$\mathcal{H}(\hat{Y})$	$\mathcal{H}(\hat{Y}_{g_1})$	$\mathcal{H}(\hat{Y}_{g_2})$	$\mathcal{H}(\hat{Y}_{g_3})$	$\mathcal{H}(\hat{Y}_{g_4})$	$\mathcal{H}(\hat{Y})$	$\mathcal{H}(P)$
scp(B,60,600)	0.8505	0.8500	0.8500	**0.8512**	**0.8512**	0.8505	0.8500	0.8500	**0.8508**	**0.8512**	0.8521
scp(C,60,600)	**0.8742**	0.8718	0.8718	**0.8742**	**0.8742**	0.8742	0.8718	0.8718	0.8742	**0.8742**	0.8832
scp(D,60,600)	**0.8726**	0.8718	0.8611	0.8621	**0.8751**	0.8726	0.8718	0.8718	**0.8742**	**0.8751**	0.8851
scp(A,80,800)	**0.8498**	0.8497	0.8497	**0.8498**	**0.8498**	0.8497	0.8497	0.8497	**0.8498**	**0.8498**	0.8502
scp(B,80,800)	**0.8535**	0.8534	0.8534	**0.8535**	**0.8535**	0.8535	0.8535	0.8535	**0.8537**	**0.8538**	0.8540
scp(C,80,800)	**0.96533**	0.9635	0.9635	0.9635	**0.96533**	**0.9635**	**0.9635**	**0.9635**	**0.9635**	**0.9635**	0.9794
scp(D,80,800)	**0.9794**	**0.9794**	**0.9794**	**0.9794**	**0.9794**	**0.9794**	**0.9794**	**0.9794**	**0.9794**	**0.9794**	0.9878
scp(A,100,1000)	0.9409	0.9342	0.9342	**0.9421**	**0.9426**	0.9409	0.9342	0.9342	**0.9421**	**0.9426**	0.9490
scp(B,100,1000)	0.9285	0.9284	0.9284	**0.9304**	**0.9310**	0.9285	0.9284	0.9284	**0.9304**	**0.9310**	0.9347

Bold indicates the best measure

Table 3 Summary of $\mathcal{H}$ -measures for $L = 5$ and $L = 7$

	Rule	$L = 5$		$L = 7$	
		Better performing number & percentage (%)		Better performing number & percentage (%)	
	9_1	11	34.38	10	31.25
$b_i \in [1,3],\ \forall\, i \in I$	9_2	4	12.50	3	9.38
Number of test instances = 32	9_3	4	12.50	3	9.38
	9_4	26	81.25	27	84.38
	9_1	13	59.09	10	45.45
$b_i = 1,\ \forall\, i \in I$	9_2	5	22.73	6	27.27
Number of test instances = 22	9_3	5	22.73	6	27.27
	9_4	18	81.82	21	95.45
	9_1	24	44.44	20	37.04
	9_2	9	16.67	9	16.67
Number of test instances = 54	9_3	9	16.67	9	16.67
	9_4	44	81.48	48	88.89

To demonstrate the LOLS algorithm's computational efficiency, we compared the observed computational times of the LOLS algorithm with existing publications in LB-AI [47], 2PPLS [29], TPM [37], and PMA [22], averaged over types of the MOSCP test instances. The study proposed in [29] provides a table for average computational time comparison for some MOSCP test problems for 2PPLS, TPM, and PMA algorithms. The study proposed in [47] has extended the same table, including LB-AI algorithms computational time. Here, we used the same table format and included the computational time of the LOLS algorithm. As mentioned in [22], we also emphasize that the computational time depends on the computer type, programming language, and implementation style. Table 5 provides this comparison for the average computational times. For example in Table 5, row corresponding to SCP(–,10,100) provides the average time for the test instances of SCP(A,10,100), SCP(B,10,100), SCP(C,10,100) and SCP(D,10,100), respectively. The last two rows of this table provide the computer type. Note that the computational times of LOLS, LB-AI, 2PPLS algorithms are collectively better than the computational times of TPM and PMA. We also note that TPM has been executed on a Pentium IV with 1.8 GHz CPUs 512 MB of RAM [37], and PMA on a computer with 400 MHz [22]. Those computers are slower than the computers used in our study, LB-AI (I-3 processor and 2 GB RAM [47] and 2PPLS (Pentium IV with 3 GHz CPUs [29]). We observe that the LOLS algorithm performs computationally well when comparing these computational times.

Table 4 Average computational time using the four classes of increasing problem size over search length

Range of n	$b_i = 1,\ \forall\, i \in I$								$b_i \in [1,3],\ \forall\, i \in I$							
	$L = 5$				$L = 7$				$L = 5$				$L = 7$			
	g_1	g_2	g_3	g_4	g_1	g_2	g_3	g_4	g_1	g_2	g_3	g_4	g_1	g_2	g_3	g_4
100–200	10.11	9.55	10.01	11.54	12.55	11.45	12.55	14.55	47.66	40.32	43.56	51.34	107.66	114.32	133.38	1541.34
400–600	21.03	20.55	19.05	21.98	24.56	23.87	22.77	24.11	55.77	54.41	55.26	60.34	125.77	126.41	125.26	133.34
800–1000	24.78	23.08	24.88	25.06	30.55	32.33	31.98	34.32	102.55	121.55	122.54	131.45	222.34	222.66	224.44	234.55
1500–2000	–	–	–	–	–	–	–	–	180.23	175.34	161.09	192.55	242.31	272.34	274.42	283.31

Table 5 Comparison of LOLS, LB-AI, 2PPLS, TPM, PMA algorithms average run times

Name	Size ($m \times n$)	LOLS	LB-AI	2PPLS	TPM	PMA
scp(–,10,100)	10×100	2.10	1.90	0.56	2.05	4.20
scp(–,40,200)	40×200	11.21	9.43	7.62	7.69	20.30
scp(–,40,400)	40×400	23.45	27.39	8.56	8.56	45.70
scp(–,60,600)	60×600	6.32	7.01	3.95	20.35	98.80
scp(–,80,800)	80×800	52.34	61.07	12.26	30.22	165.9
scp(–,100,1000)	100×1000	30.23	33.33	13.76	50.10	311.70
Computer		I-7 processor	I-3 processor	Pentium IV	Pentium IV	Computer
RAM		8 GB	2 GB	3 GHz	1.8 GHz 512 MB	400 MHz

5.4 Comparison with a state-of-the-art algorithm for the MOSCP

We compare the performance of the LOLS algorithm to LB-AI [47], 2PPLS [29], TPM [37], and PMA [21] based on $\mathcal{H}$ measures for the MOSCP. Here, we present the $\mathcal{H}$ measure over nonscaled Pareto points for this comparison. We consider various $scp(l, m, n)$ test problems, and Table 6 provides the corresponding measures. It can be observed that the LOLS algorithm significantly outperforms LB-AI on many instances in terms of $\mathcal{H}$ measure. One of the main differences between the LOLS algorithm and the LB-AI algorithm is constructing the initial solution set. The LOLS algorithm obtains the initial solutions by solving the scalarized GMOSCPs using achievement functions with one reference point. In contrast, the LB-AI algorithm obtained the initial solutions by solving scalarized GMOSCPs using the classical weighted-sum method. From this analysis, we observe that the LOLS algorithm significantly outperforms LB-AI on many instances of the $\mathcal{H}$ measure. This performance illustrates the advantage of integrating the achievement scalarization with a reference point over the classical weighted sum method in the partitioning algorithm proposed in [11]. We also observe that the LOLS algorithm outperforms TPM and PMA all the time and performs better on five test problems out of ten test problems compared to 2PPLS that have been considered in this study based on $\mathcal{H}$ measure.

5.5 Comparison with the Pareto set

We analyzed the performance metrics produced by the LOLS algorithm using the Pareto sets. We computed the exact Pareto outcomes by applying the ϵ-constraint method [15]. We note that improved techniques for computing Pareto sets have been studied in recent literature [12, 33, 35, 50]. The nonscaled $\mathcal{H}$ measures for the benchmark test problems are also provided in [12, 29, 33, 47]. In addition, we provided the scaled (normalized) $\mathcal{H}$ measures for the Pareto sets in the last column of Table 2 and the nonscaled (non-normalized) $\mathcal{H}$ measures in the last column of Table 6 of the LOLS algorithm regarding the Pareto sets. By definition, the $\mathcal{H}$ measure of Y_P is always larger than that of $\hat{Y}$. Based on the numerical data provided in Table 6, we observed that the LOLS algorithm performs well, and it indicates

Table 6 Comparisons with state-of-the-art algorithm on various *scp*(*l*, *m*, *n*) instances

Instances	$\mathcal{H}(\hat{Y}_{g_1})$	$\mathcal{H}(\hat{Y}_{g_2})$	$\mathcal{H}(\hat{Y}_{g_3})$	$\mathcal{H}(\hat{Y}_{g_4})$	State-of-the-Art					
					LOLS	LB-AI	2PPLS	TPM	PMA	$\mathcal{H}(Y_P)$
scp(A,10,100)	1.6538	1.6538	1.6538	1.6538	**1.6538**	1.6502	–	–	–	1.6559
scp(B,10,100)	1.0360	1.0360	1.0360	1.0362	1.0362	**1.0370**	–	–	–	1.0384
scp(C,10,100)	2.3950	2.3950	2.3950	2.3979	**2.3979**	0.1703	–	–	–	0.2400
scp(D,10,100)	0.2793	0.2793	0.2793	0.2793	**0.2793**	0.2573	–	–	–	0.2793
scp(A,40,200)	31.4393	31.4392	31.4392	31.4340	**31.4400**	31.4030	–	–	–	31.4500
scp(B,40,200)	51.3735	51.3625	51.3625	51.3720	**51.3751**	51.3690	–	–	–	51.3880
scp(C,40,200)	41.8596	41.8596	41.8596	41.8596	41.8596	**42.2530**	–	–	–	42.4300
scp(D,40,200)	12.2184	12.3576	12.3576	12.3626	12.3626	**12.3880**	–	–	–	12.4330
scp(A,40,400)	145.9662	145.9651	145.9651	145.9672	**145.9734**	145.8300	–	–	–	146.0200
scp(B,40,400)	272.9507	272.8965	272.8965	272.9751	**272.9884**	272.7600	–	–	–	273.0900
scp(C,40,400)	156.3073	156.2305	156.2305	156.3931	**156.4175**	155.9300	–	–	–	157.3400
scp(D,40,400)	9.9224	9.9224	9.9224	9.9224	**9.9224**	9.8290	–	–	–	9.9992
scp(A,60,600)	650.0121	649.5129	649.5129	650.2876	**650.3191**	649.3400	650.4600	646.9000	639.06	650.6800
scp(B,60,600)	806.4309	806.0459	806.0459	806.0459	806.6595	805.1900	**806.8300**	801.0800	784.67	807.5900
scp(C,60,600)	77.5404	77.4080	77.4080	77.5404	**77.5404**	77.4040	76.9300	72.8100	66.19	78.6800
scp(D,60,600)	584.0282	584.0282	583.6222	584.8916	585.3125	**588.2**	585.3000	581.5900	549.45	590.4600
scp(A,80,800)	1804.9645	1804.9373	1804.9373	1805.0667	**1805.1695**	1803.68	1804.75	1787.61	1777.71	1805.7870
scp(B,80,800)	2296.6585	2296.6982	2296.6982	2296.9756	**2297.1529**	2294.07	2295.73	2277.46	2267.98	2297.6619
scp(C,80,800)	112.9589	112.9589	112.9589	112.9589	112.9589	**114.0000**	113.91	114.5	110.06	114.4498
scp(D,80,800)	225.2692	225.2692	225.2692	225.2692	**225.2692**	169.56	168.54	169.4	160	226.7785
scp(A,100,1000)	304.0900	302.3400	302.3400	304.4200	304.5600	305.0900	**305.8500**	286.7300	305.2000	306.2192
scp(B,100,1000)	338.6400	338.6200	338.6200	339.2000	339.4000	339.3200	**340.0900**	329.3600	338.9000	340.4820

Bold indicates the best measure

Table 7 $\mathcal{C}$- metric comparisons with the Pareto set when $L = 7$

Instances	$\mathcal{C}(\hat{Y}_{g_1}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_2}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_3}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_4}, Y_p)$	$\mathcal{C}(\hat{Y}, Y_p)$	Instances	$\mathcal{C}(\hat{Y}_{g_1}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_2}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_3}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_4}, Y_p)$	$\mathcal{C}(\hat{Y}, Y_p)$
gscp(A,10,100)	0.2740	0.3014	0.3014	0.3425	0.4384	scp(A,10,100)	0.6500	0.6500	0.6500	0.7250	0.7250
gscp(B,10,100)	0.4211	0.2281	0.2281	0.4737	0.4737	scp(B,10,100)	0.6829	0.6341	0.6341	0.7073	0.7805
gscp(C,10,100)	0.6667	0.6667	0.6667	0.7143	0.7143	scp(C,10,100)	0.6667	0.5556	0.5556	0.7778	0.7778
gscp(D,10,100)	0.6667	0.6667	0.6667	0.6667	0.6667	scp(D,10,100)	1.0000	1.0000	1.0000	1.0000	1.0000
gscp(A,40,200)	0.0840	0.0840	0.0840	0.0840	0.0840	scp(A,40,200)	0.7757	0.7664	0.7664	0.7383	0.7850
gscp(B,40,200)	0.0662	0.0662	0.0662	0.0662	0.0662	scp(B,40,200)	0.7982	0.7798	0.7798	0.8257	0.8440
gscp(C,40,200)	0.1887	0.1887	0.1887	0.1887	0.1887	scp(C,40,200)	0.6957	0.6957	0.6957	0.6957	0.6957
gscp(D,40,200)	0.1667	0.1667	0.1667	0.1667	0.1667	scp(D,40,200)	0.3115	0.2951	0.2951	0.3115	0.3115
gscp(A,40,400)	0.0313	0.0313	0.0313	0.0313	0.0313	scp(A,40,400)	0.7729	0.7536	0.7536	0.7681	0.7971
gscp(B,40,400)	0.0262	0.0262	0.0262	0.0314	0.0314	scp(B,40,400)	0.5663	0.5233	0.5233	0.6022	0.6452
gscp(C,40,400)	0.1189	0.0909	0.0909	0.0979	0.1189	scp(C,40,400)	0.3617	0.3511	0.3511	0.3617	0.3936
gscp(D,40,400)	0.6111	0.6111	0.6111	0.6111	0.6111	scp(D,40,400)	0.5333	0.5333	0.5333	0.5333	0.5333
gscp(A,60,600)	0.0274	0.0274	0.0274	0.0324	0.0324	scp(A,60,600)	0.4180	0.3438	0.3438	0.5352	0.5586
gscp(B,60,600)	0.0217	0.0217	0.0217	0.0236	0.0236	scp(B,60,600)	0.4357	0.3567	0.3567	0.5088	0.5614
gscp(C,60,600)	0.2553	0.2553	0.2553	0.2553	0.2553	scp(C,60,600)	0.5000	0.4643	0.4643	0.6071	0.6071
gscp(D,60,600)	0.0893	0.0893	0.0893	0.0893	0.0893	scp(D,60,600)	0.2917	0.2917	0.2917	0.3750	0.3750
gscp(A,80,800)	0.0142	0.0142	0.0142	0.0142	0.0142	scp(A,80,800)	0.9649	0.3646	0.3646	0.4479	0.5104
gscp(B,80,800)	0.0171	0.0171	0.0171	0.0171	0.0171	scp(B,80,800)	0.3600	0.2900	0.2900	0.4600	0.5400
gscp(C,80,800)	0.1026	0.1026	0.1026	0.1026	0.1026	scp(C,80,800)	0.8571	0.8571	0.8571	0.8571	0.8571
gscp(D,80,800)	0.4118	0.4118	0.4118	0.4118	0.4118	scp(D,80,800)	0.2564	0.2564	0.2564	0.4615	0.4615
gscp(A,100,1000)	0.0418	0.0418	0.0418	0.0418	0.0418	scp(A,100,1000)	0.1132	0.1069	0.1069	0.1069	0.1132
gscp(B,100,1000)	0.0473	0.0473	0.0473	0.0473	0.0473	scp(B,100,1000)	0.1119	0.0979	0.0979	0.1259	0.1295
rscp(20,500)	0.5385	0.5000	0.5000	0.5385	0.5385						
rscp(75,600)	0.3659	0.3171	0.3171	0.3659	0.3659						
rscp(20,700)	0.5000	0.4545	0.4545	0.5000	0.5000						
rscp(100,750)	0.4103	0.3846	0.3846	0.4103	0.4103						

that the $\mathcal{H}$ measures are reasonable. Table 7 provides the comparison based on the $\mathcal{C}$ metric. We showed the results for $\mathcal{C}(\hat{Y}_{g_i}, Y_p)$, $i = 1, 2, 3, 4$, and for $\hat{Y}$ when $L = 7$. We do not show the results for $\mathcal{C}(Y_p, \hat{Y}_{g_i})$ since always $\mathcal{C}(Y_p, \hat{Y}_{g_i}) = 1$ for $i = 1, 2, 3, 4$. Specifically, we note that for the MOSCP cases, we have $\mathcal{C}(Y_p, \hat{Y}_{g_i}) > 0.5$ for 18 test cases out of 22 test cases. This means the LOLS algorithm finds more than 50% of Pareto points for most test problems. For the GMOSCP test problems we have $\mathcal{C}(Y_p, \hat{Y}_{g_i}) > 0.4$ for 10 test cases and for other test cases $\mathcal{C}(Y_p, \hat{Y}_{g_i}) < 0.4$. The GMOSCP is a challenging problem compared to the MOSCP due to the multi-cover constraints, thus difficult to obtain the majority of the Pareto points. Though, compared with $\mathcal{H}$ measures provided in Tables 2 and 6, we observe $\mathcal{H}(\hat{Y})$ and $\mathcal{H}(Y_p)$ are very close, which implies that the points in the two sets $\hat{Y}$ and Y_p are very close to each other. Thus, we conclude that the LOLS algorithm performs well on the test problems studied here.

6 Conclusion and future work

This paper proposes a novel algorithm, called the LOLS algorithm, that is based on the lexicographic ordering set selection approach to approximating the Pareto set of the GMOSCP. The algorithm is integrated into the branching of the feasible region algorithm proposed in [11] to solve single-objective mixed-integer optimization problems. In LOLS, the distribution of the initial population can be better maintained by abating and solving the scalarized GMOSCPs using achievement functions with a reference point. We investigated the performance of the LOLS algorithm in detail. We considered various test instances from the literature, and we also proposed a method for generating random test instances.

We introduced four different cost-efficient rules to convert partial (infeasible) solutions to feasible solutions. We investigated whether there is a better relationship between cost-efficient value and set density other than the classical linear relationship for the GMOSCP. The cost-efficient rule g_1 is a linear fractional rule while the cost-efficient rules g_2, g_3, g_4 are non-linear fractional rules. With the cost-efficient rule g_4, we prioritized the cost-value considering a non-linear fractional term. Thus, in contrast to cost-efficient rules g_1, g_2, g_3, we dicovered that the rule g_4 performed well on the GMOSCP. We discovered that having reasonably well multiple priority rules in the LOLS algorithm results in a better approximation for the GMOSCP. In addition, we investigated a better-performing cost-efficient rule for the GMOSCP among the four cost-efficient rules proposed in the study. The GMOSCP can be reduced to MOSCP if $b_i \geq 1$ for all $i \in I$. Thus, we compared the performance of the LOLS algorithm with the state-of-the-art MOSCP algorithms. This comparison shows that the LOLS algorithm significantly outperforms the LB-AI algorithm, the most recent algorithm available in the literature to solve the MOSCP on many problem instances in terms of the $\mathcal{H}$ measure. One of the main differences between the LOLS algorithm and the LB-AI algorithm is constructing the initial solution set. The LOLS algorithm obtains the initial solutions by solving the scalarized GMOSCPs using achievement functions while the LB-AI algorithm obtains the initial solutions by solving scalarized GMOSCPs using the classical weighted-sum

Table 7 (continued)

Instances	$\mathcal{C}(\hat{Y}_{g_1}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_2}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_3}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_4}, Y_p)$	$\mathcal{C}(\hat{Y}, Y_p)$	Instances	$\mathcal{C}(\hat{Y}_{g_1}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_2}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_3}, Y_p)$	$\mathcal{C}(\hat{Y}_{g_4}, Y_p)$	$\mathcal{C}(\hat{Y}, Y_p)$
rscp(70,900)	0.4138	0.3448	0.3448	0.4138	0.4138						
rscp(100,1000)	0.3023	0.2558	0.2558	0.3023	0.3023						
rscp(250,1000)	0.2653	0.2462	0.2462	0.2463	0.2653						
rscp(100,1500)	0.3889	0.4167	0.4167	0.3889	0.4167						
rscp(200,1500)	0.1897	0.1897	0.1897	0.1897	0.1897						
rscp(200,2000)	0.2721	0.2708	0.2708	0.2708	0.2721						

method. Comparing the LOLS algorithm with LB-AI, 2PPL, TPM, and PMA shows that the LOLS algorithm is more efficient.

The following are possible areas for further investigation: Investigating other lexicographic cost-selection rules that benefit the GMOSCP is also worth exploring. Also, an ANOVA-type analysis can be used to compare approximations of the Pareto set based on each cost-efficient rule.

Acknowledgements We are very grateful to all reviewers for the careful analysis of our text and valuable feedback.

Data availability We have used two sets of data sets. The first set of data used in this study has been previously published. This set of data has also been used in our citations 13,35,37,53. The data set can be accessed using the following link https://github.com/vOptSolver/vOptLib/blob/master/SCP/readme.md. We have described the procedure for generating the second set of data in the manuscript.

References

1. Alsheddy, A., Tsang, E.E.: Guided pareto local search based frameworks for biobjective optimization. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2010)
2. Bandara, D., Mayorga, M., McLay, M.L.: Optimal dispatching strategies for emergency vehicles to increase patient survivability. Int. J. Oper. Res. **15**(2), 195–214 (2012)
3. Bettinelli, A., Ceselli, A., Righini, G.: A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. Math. Program. Comput. **6**(2), 171–197 (2014)
4. Chvatal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**(3), 233–235 (1979)
5. Czyzżak, P., Jaszkiewicz, A.: Pareto simulated annealing-a metaheuristic technique for multiple-objective combinatorial optimization. J. Multi-criteria Decis. Anal. **7**(1), 34–47 (1998)
6. Daskin, M.S., Stern, E.H.: A hierarchical objective set covering model for emergency medical service vehicle deployment. Transp. Sci. **15**(2), 137–152 (1981)
7. Ehrgott, M.: Approximation algorithms for combinatorial multicriteria optimization problems. Int. Trans. Oper. Res. **7**(1), 5–31 (2000)
8. Ehrgott, M.: Multicriteria Optimization. Springer, New York (2006)
9. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum **22**(4), 425–460 (2000)
10. Figueira, J.R., Liefooghe, A., Talbi, E.-G., Wierzbicki, A.P.: A parallel multiple reference point approach for multi-objective optimization. Eur. J. Oper. Res. **205**(2), 390–400 (2010)
11. Fischetti, M., Lodi, A.: Local branching. Math. Program. **98**(1–3), 23–47 (2003)
12. Florios, K., Mavrotas, G.: Generation of the exact pareto set in multi-objective traveling salesman and set covering problems. Appl. Math. Comput. **237**, 1–19 (2014)
13. Gandibleux, X., Mezdaoui, N., Fréville, A.: A tabu search procedure to solve multiobjective combinatorial optimization problems. In: Advances in Multiple Objective and Goal Programming, pp. 291–300. Springer, New York (1997)
14. García-Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp. Eur. J. Oper. Res. **180**(1), 116–148 (2007)
15. Haimes, Y.: On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Trans. Syst. Man Cybern. **1**(3), 296–297 (1971)
16. Hammer, P.L., Bonates, T.O.: Logical analysis of data: an overview: from combinatorial optimization to medical applications. Ann. Oper. Res. **148**(1), 203–225 (2006)
17. Hamming, R.W.: Error detecting and error correcting codes. Bell Syst. Tech. J. **29**(2), 147–160 (1950)
18. Hansen, M.P.: Use of substitute scalarizing functions to guide a local search based heuristic: the case of MOTSP. J. Heuristics **6**(3), 419–431 (2000)
19. https://github.com/vOptSolver/vOptLib/tree/master/SCP

20. Jaszkiewicz, A.: Genetic local search for multi-objective combinatorial optimization. Eur. J. Oper. Res. **137**(1), 50–71 (2002)
21. Jaszkiewicz, A.: Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem. IEEE Trans. Evol. Comput. **7**(2), 133–143 (2003)
22. Jaszkiewicz, A.: A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm. Ann. Oper. Res. **131**(1–4), 135–158 (2004)
23. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, pp. 85–103. Springer (1972)
24. Ke, L., Zhang, Q., Battiti, R.: Moea/d-aco: a multiobjective evolutionary algorithm using decomposition and antcolony. IEEE Trans. Cybern. **43**(6), 1845–1859 (2013)
25. Kohl, N., Karisch, S.E.: Airline crew rostering: problem types, modeling, and optimization. Ann. Oper. Res. **127**(1–4), 223–257 (2004)
26. Lan, G., DePuy, G.W., Whitehouse, G.E.: An effective and simple heuristic for the set covering problem. Eur. J. Oper. Res. **176**(3), 1387–1403 (2007)
27. Liang, Y.-C., Lo, M.-H.: Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm. J. Heuristics **16**(3), 511–535 (2010)
28. Lust, T., Teghem, J., Tuyttens, D.: Very large-scale neighborhood search for solving multiobjective combinatorial optimization problems. In: International Conference on Evolutionary Multi-Criterion Optimization, pp. 254–268. Springer (2011)
29. Lust, T., Tuyttens, D.: Variable and large neighborhood search to solve the multiobjective set covering problem. J. Heuristics **20**(2), 165–188 (2014)
30. Marchiori, E., Steenbeek, A.: An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In: Workshops on Real-World Applications of Evolutionary Computation, pp. 370–384. Springer (2000)
31. Marchiori. E., Steenbeek, A.: An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In: 41st Annual Symposium on Real-World Applications of Evolutionary Computation, Workshops, pp. 370–384. Springer, Berlin (2000)
32. Marsten, R.E., Shepardson, F.: Exact solution of crew scheduling problems using the set partitioning model: recent successful applications. Networks **11**(2), 165–177 (1981)
33. Mavrotas, G., Florios, K.: An improved version of the augmented $\varepsilon $-constraint method (augmecon2) for finding the exact pareto set in multi-objective integer programming problems. Appl. Math. Comput. **219**(18), 9652–9669 (2013)
34. McDonnell, M.D., Possingham, H.P., Ball, I.R., Cousins, E.A.: Mathematical methods for spatially cohesive reserve design. Environ. Model. Assess. **7**(2), 107–114 (2002)
35. Nikas, A., Fountoulakis, A., Forouli, A., Doukas, H.: A robust augmented $\varepsilon $-constraint method (augmecon-r) for finding exact solutions of multi-objective linear programming problems. Oper. Res. 1–42 (2020)
36. Paquete, L., Stützle, T.: Design and analysis of stochastic local search for the multiobjective traveling salesman problem. Comput. Oper. Res. **36**(9), 2619–2631 (2009)
37. Prins, C., Prodhon, C., Calvo, R.W.: Two-phase method and Lagrangian relaxation to solve the bi-objective set covering problem. Ann. Oper. Res. **147**(1), 23–41 (2006)
38. Revelle, C., Hogan, K.: The maximum reliability location problem and $\alpha $-reliablep-center problem: derivatives of the probabilistic location set covering problem. Ann. Oper. Res. **18**(1), 155–173 (1989)
39. Saxena, R.R., Arora, S.R.: Exact solution of crew scheduling problems using the set partitioning model: recent successful applications. Optimization **11**(2), 165–177 (1981)
40. Soylu, B.: Heuristic approaches for biobjective mixed 0–1 integer linear programming problems. Eur. J. Oper. Res. **245**(3), 690–703 (2015)
41. Steuer, R.E.: Multiple criteria optimization. Theory Comput. Appl. (1986)
42. Ulungu, E.L., Teghem, J.: Multi-objective combinatorial optimization problems: a survey. J. Multicriteria Decis. Anal. **3**(2), 83–104 (1994)
43. Vasko, F.J.: An efficient heuristic for large set covering problems. Naval Res. Logist. Q. **31**(1), 163–171 (1984)
44. Weerasena, L.: Algorithm for generalised multi-objective set covering problem with an application in ecological conservation. Int. J. Math. Model. Numer. Optim. **10**(2), 167–186 (2020)
45. Weerasena, L., Shier, D., Tonkyn, D.: A hierarchical approach to designing compact ecological reserve systems. Environ. Model. Assess. **19**(5), 437–449 (2014)

46. Weerasena, L., Wiecek, M.M.: A tolerance function for the multiobjective set covering problem. Optim. Lett. 1–19 (2018)
47. Weerasena, L., Wiecek, M.M., Soylu, B.: An algorithm for approximating the pareto set of the multiobjective set covering problem. Ann. Oper. Res. **248**(1–2), 493–514 (2017)
48. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Multiple Criteria Decision Making Theory and Application, pp. 468–486. Springer (1980)
49. Wierzbicki, A.P.: On the completeness and constructiveness of parametric characterizations to vector optimization problems. Oper. Res. Spektrum **8**(2), 73–87 (1986)
50. Zhang, W., Reimann, M.: A simple augmentedâ-constraint method for multi-objective mathematical integer programming problems. Eur. J. Oper. Res. **234**(1), 15–24 (2014)
51. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000)