Computing the nadir point for multiobjective discrete optimization problems

Article *in* Journal of Global Optimization · May 2015

DOI: 10.1007/s10898-014-0227-6

2 authors, including:

Gokhan Kirlik
University of Maryland Medical System, Baltimore, United States
19 PUBLICATIONS **228** CITATIONS

SEE PROFILE

J Glob Optim
DOI 10.1007/s10898-014-0227-6

Computing the nadir point for multiobjective discrete optimization problems

Gokhan Kirlik · Serpil Sayın

Received: 28 May 2013 / Accepted: 25 July 2014

Abstract We investigate the problem of finding the nadir point for multiobjective discrete optimization problems (MODO). The nadir point is constructed from the worst objective values over the efficient set of a multiobjective optimization problem. We present a new algorithm to compute nadir values for MODO with p objective functions. The proposed algorithm is based on an exhaustive search of the $(p-2)$-dimensional space for each component of the nadir point. We compare our algorithm with two earlier studies from the literature. We give numerical results for all algorithms on multiobjective knapsack, assignment and integer linear programming problems. Our algorithm is able to obtain the nadir point for relatively large problem instances with up to five-objectives.

Keywords Nadir point · Multiobjective optimization · Multiobjective discrete optimization

1 Introduction

Optimization algorithms are typically developed with the purpose of optimizing a single objective. However, many real-world applications require considering multiple objectives simultaneously. Any optimization problem with more than one objective function is called a multiobjective optimization problem (MOP), which no longer possesses a unique optimal objective function value. In multiobjective optimization, the set of efficient solutions is used instead of the optimal solution. An efficient solution has the property that no improvement in any objective is possible without sacrificing in at least one other objective. The set of all efficient solutions is called the efficient set. This set portrays all relevant trade-off information to a decision maker.

G. Kirlik
Graduate School of Sciences and Engineering, Koç University, Sariyer, 34450 Istanbul, Turkey
e-mail: gkirlik@ku.edu.tr

S. Sayın (✉)
College of Administrative Sciences and Economics, Koç University, Sariyer, 34450 Istanbul, Turkey
e-mail: ssayin@ku.edu.tr

Published online: 07 August 2014

In this study, we investigate the determination of the nadir point for multiobjective discrete optimization (MODO) problems. MODO is a special case of MOP where all variables are discrete. The nadir point consists of worst objective values attained over the efficient set. Obtaining the nadir point is generally a hard problem [17]. Along with the relatively easy to obtain ideal point, the nadir point is an important element of MOP, because these points define lower and upper bounds of the efficient set. In fact, there are some methods that require the nadir point as input, especially among interactive approaches such as in [18,24,27]. Hence, determination of the nadir point has been studied extensively and several exact and heuristic methods have been proposed for the problem [17].

Earlier studies on the nadir point were proposed for multiobjective linear programming (MOLP) problems. Isermann and Steuer [21] propose three different approaches to compute the nadir point for MOLP. The first obtains the nadir point after computing all efficient solutions. The second solves a large primal-dual feasible program with nonlinear constraints. The third is a simplex-based procedure using the fact that the efficient extreme points are connected by efficient edges. Recently, Alves and Costa [2] present a method to compute the nadir point for MOLP by using weight space search. This method determines, for each objective function, the region of the weight space associated with the efficient solutions that have a value in that criterion worse than already known. A new efficient solution is computed with weighted sum formulation using a weight vector picked from the region. The search continues until the region is empty.

Obtaining the nadir point is a special case of the problem of optimization over the efficient set. This is a global optimization problem [5] and has been addressed in several studies. Several of these studies consider the optimization of a linear function over the efficient set of a MOLP, for instance as in [5–9,14,15,30]. While [13] considers a nonlinear function [10], takes into account the minimization of a quasi-concave function over the efficient set of a MOLP. Maximization of convex, concave and quadratic functions over the efficient set of a convex mathematical program is presented in [3]. Recently, [11] presents a method that minimizes a convex function over the efficient set of a convex MOP. Survey of existing algorithms for optimization over the efficient set is given in [31]. Although these procedures are theoretically able to compute the nadir values, they are rather complex algorithms with limited or no computational verification [17]. Generally, optimization over efficient set studies consider convex MOP efficient sets. As exceptions, [1] and [22] optimize a linear function over a multiobjective integer efficient set.

For the MODO problem, the most naive way to compute the nadir point is to obtain the entire efficient set. This is computationally demanding, and is not appropriate, for instance, for algorithms that require the nadir point as an input to compute the entire efficient set. Ehrgott and Tenfelde-Podehl [17] present an important result to compute the nadir point for MOP in a computationally more affordable way. The method removes one objective function from the objective functions vector and enumerates all efficient solutions of the remaining MOP with $(p-1)$ objective functions. After repeating for each objective function, these solutions are merged into one set. Some solutions in this set that may not be efficient for the original MOP are eliminated. Finding the supremum of the set for each objective function results in the nadir point for MOP. This method requires an algorithm that generates the efficient set of MOP with $(p-1)$ objective functions and computational verification of the algorithm is not reported in [17].

Another way to obtain the nadir point for MODO problems is to use optimization over integer efficient set algorithms. As an example, in [1], different types of cuts are imposed to improve the objective value at each iteration. The algorithm by Jorge [22] is based on solving more constrained integer linear programs progressively. Both algorithms rely on

adding cuts to the single-objective optimization problem, and avoid explicit enumeration of the efficient set. However, depending on the instance, the additional cuts may make the problem computationally overwhelming.

In this study, we present a new algorithm to compute the nadir point for MODO with any number of objective functions. In this algorithm, computation of the nadir value is achieved by an exhaustive search in $(p-2)$-dimensional space which contains projection of all nondominated solutions onto $\mathbb{R}^{p-2}$. We partition this space into $(p-2)$-dimensional rectangles to search the space entirely. During the search, a two-stage ε-constraint scalarization is used to obtain nondominated solutions. The method guarantees to find the nadir point in a finite number of iterations.

In the following section, we present the notation and some definitions related to MOP. In Sect. 3, the theoretical basis for our approach is introduced. In Sect. 4, our algorithm to compute the nadir point for MODO is given. In Sect. 5, computational results for our algorithm, Ehrgott and Tenfelde-Podehl's algorithm [17], and Jorge's algorithm [22] are given on multiobjective knapsack, assignment and integer linear programming problem instances. Finally, conclusions are presented in Sect. 6.

2 Notation and definitions

In MODO, p objective functions $f_j(x) : \mathbb{R}^n \to \mathbb{R}$ for $j = 1, \ldots, p$ have to be minimized. The feasible set is discrete and is denoted as $\mathcal{X} \subseteq \mathbb{Z}^n$. Each feasible solution $x \in \mathcal{X}$ is mapped into its corresponding objective vector $y = f(x)$ and $\mathcal{Y} = \{y \in \mathbb{R}^p : y = f(x) \text{ for some } x \in \mathcal{X}\}$ is referred to as the set of feasible outcomes in the objective space. In mathematical terms, MODO is defined as:

$$\begin{aligned} (\textbf{MODO}) \ \min \ & f(x) = [f_1(x), \ldots, f_p(x)] \\ \text{s.t.} \ & x \in \mathcal{X} \end{aligned}$$

Due to conflicting objectives, MODO is expected to have more than one solution. These solutions are called efficient solutions.

Definition 1 A solution $x^* \in \mathcal{X}$ is called an *efficient solution* if there exists no feasible solution $x \in \mathcal{X}$ such that $f_j(x) \leq f_j(x^*)$ for all $j = \{1, \ldots, p\}$ and there exists $\hat{j} \in \{1, \ldots, p\}$ such that $f_{\hat{j}}(x) < f_{\hat{j}}(x^*)$. For an efficient solution x^*, $f(x^*) \in \mathbb{R}^p$ is referred to as a *nondominated solution* in the outcome space.

The set of all efficient solutions for MODO is called the efficient set and is denoted as $\mathcal{X}_E$. The image of the efficient set in the objective space is called the nondominated set and is denoted as $\mathcal{Y}_{ND}$, i.e. $\mathcal{Y}_{ND} = \{y \in \mathbb{R}^p : y = f(x) \text{ for some } x \in \mathcal{X}_E\}$.

Throughout the paper we assume that the efficient set $\mathcal{X}_E$ is a nonempty and bounded set. With this assumption, the ideal and nadir point, denoted as y^I and y^N respectively, determine the bounds of the nondominated set. Hence, these points satisfy $y^I_j \leq y_j \leq y^N_j$ for each $j \in \{1, \ldots, p\}$ and for all $y \in \mathcal{Y}_{ND}$. Mathematical definitions of the ideal and nadir point are given below.

Definition 2 The point $y^I = (y^I_1, \ldots, y^I_p)$ given by

$$y^I_j = \min_{x \in \mathcal{X}_E} f_j(x) = \min_{y \in \mathcal{Y}_{ND}} y_j \quad j = 1, \ldots, p \tag{1}$$

is called the ideal point.

From a multiobjective point of view, computation of the ideal point can be considered as easy [16]. It is well-known that $y_j^I = \min_{x\in\mathcal{X}_E} f_j(x) = \min_{x\in\mathcal{X}} f_j(x)$ for $j = 1, \ldots, p$.

Definition 3 The point $y^N = (y_1^N, \ldots, y_p^N)$ given by

$$y_j^N = \max_{x\in\mathcal{X}_E} f_j(x) = \max_{y\in\mathcal{Y}_{ND}} y_j \quad j = 1, \ldots, p$$

is called the nadir point.

Obtaining the nadir point is a challenging task except for the case $p = 2$. In bicriteria optimization, the worst value of the second objective function is attained among solutions that minimize the first objective function and vice versa, which makes it easy to compute the nadir point. The well-known payoff table solution [29] can be considered as a generalization of this approach for $p > 2$.

Let T represent the payoff table. This table consists of p rows and p columns and is computed by solving single criterion optimization problems. For $m \in \{1, \ldots, p\}$, let x_m^* be an optimal solution to the lexicographic optimization problem [4] where mth objective function is first in the order. Then the entry in the mth row and the kth column of the payoff table is given by $T_{mk} = f_k(x_m^*)$ for $k \in \{1, \ldots, p\}$. The payoff table estimate y_k^{PT} for the kth entry of the nadir point is obtained as follows,

$$y_k^{PT} = \max_{m=1,\ldots,p} T_{mk} = \max_{m=1,\ldots,p} f_k(x_m^*) \quad k = 1, \ldots, p. \tag{2}$$

When $p > 2$, payoff table approach can generate only a heuristic solution instead of the exact nadir point, i.e. $y_j^{PT} \leq y_j^N$ for each $j \in \{1, \ldots, p\}$. Obtaining the payoff table T requires solving p lexicographic optimization problems, i.e. p^2 single-objective optimization problems. Hence complexity of obtaining y^{PT} is $O(p^2 \cdot t)$ where t is the running time of the single-objective optimizer.

According to [2], payoff table can be a good estimate of the nadir point in MOLP problems especially when p is small. We wish to incorporate y^{PT} into our search for the nadir point. Furthermore, we provide an analysis of proximity of y^{PT} to y^N over our test instances.

3 Theoretical background

For the presentation of our approach, we use an unattainable overestimator of the nadir point which we define as $y_j^U = \max_{x\in\mathcal{X}} f_j(x) + \delta$ for $j = 1, \ldots, p$ where $\delta > 0$. Clearly, $y_j^N < y_j^U$ for $j = 1, \ldots, p$.

The ε-constraint method suggests retaining one of the p objective functions as the objective function while remaining $(p-1)$ are turned into constraints [19]. The weakly efficient solutions that may be found are then eliminated by using techniques such as lexicographic, augmented or two-stage subproblems as recently implemented in [23,25,28]. Here, we propose a slight modification of the two-stage ε-constraint method for the purpose of seeking nondominated solutions that determine the kth component of the nadir point. This is achieved by inactivating the kth objective function. For any $\varepsilon \in \mathbb{R}^{p-2}$, $P_{k,m}(\varepsilon)$ and $Q_{k,m}(\varepsilon)$ for some $k, m \in \{1, \ldots, p\}$ and $k \neq m$, are defined as follows.

$$\begin{aligned} \mathbf{P_{k,m}}(\varepsilon) \quad \mathbf{z} = \min \quad & f_m(x) \\ \text{s.t.} \quad & f_k(x) \leq y_k^U \\ & f_j(x) \leq \varepsilon_j \quad j \in C \\ & x \in \mathcal{X}. \end{aligned}$$

Above, set C is defined as $C = \{1, \ldots, p\} \setminus \{k, m\}$. Let z^* be the optimal objective value of subproblem $P_{k,m}(\varepsilon)$ and consider the second stage formulation $Q_{k,m}(\varepsilon)$.

$$\begin{aligned} \mathbf{Q_{k,m}}(\varepsilon) \quad \min \quad & \sum_{j=1}^{p} f_j(x) \\ \text{s.t.} \quad & f_m(x) = z^* \\ & f_k(x) \leq y_k^U \\ & f_j(x) \leq \varepsilon_j \quad j \in C \\ & x \in \mathcal{X}. \end{aligned}$$

Let x^* be an optimal solution to two-stage formulations $P_{k,m}(\varepsilon)$ and $Q_{k,m}(\varepsilon)$ where $k, m \in \{1, \ldots, p\}$ and $k \neq m$. That x^* is always efficient for any $\varepsilon \in \mathbb{R}^{p-2}$ follows easily from earlier results [23]. Besides, whenever $f_k(x^*) = y_k^N$, there exist an $\varepsilon \in \mathbb{R}^{p-2}$ such that x^* is optimal to two-stage programs.

Theorem 1 *Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$. For $\varepsilon \in \mathbb{R}^{p-2}$, an optimal solution to the two-stage formulations $P_{k,m}(\varepsilon)$ and $Q_{k,m}(\varepsilon)$ is efficient.*

Theorem 2 *Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$. For an efficient solution x^* where $f_k(x^*) = y_k^N$, there exists an $\varepsilon \in \mathbb{R}^{p-2}$ such that x^* is an optimal solution to two-stage formulations $P_{k,m}(\varepsilon)$ and $Q_{k,m}(\varepsilon)$.*

Proof Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$. Suppose x^* is a feasible solution such that $f_k(x^*) = y_k^N$. Set $\hat{\varepsilon}_j = f_j(x^*)$ for each $j \in C$. Note that x^* is feasible to $P_{k,m}(\hat{\varepsilon})$. Suppose that x^* does not solve two-stage formulations $P_{k,m}(\hat{\varepsilon})$ and $Q_{k,m}(\hat{\varepsilon})$. Let x' be an optimal solution to two-stage programs with $\hat{\varepsilon} \in \mathbb{R}^{p-2}$. Then either $f_m(x') < f_m(x^*)$ or $f_m(x') = f_m(x^*)$ and $\sum_{j=1}^{p} f_j(x') < \sum_{j=1}^{p} f_j(x^*)$. Since $f_j(x') \leq \hat{\varepsilon}_j = f_j(x^*)$ for all $j \in \{1, \ldots, p\} \setminus \{k\}$ and $f_k(x') \leq f_k(x^*) = y_k^N$, $f_{\hat{j}}(x') < f_{\hat{j}}(x^*)$ for some $\hat{j} \in \{1, \ldots, p\}$. This implies that x' dominates x^*, which contradicts that x^* is an efficient solution. □

Remark 1 Theorem 2 shows that y_k^N, $k \in \{1, \ldots, p\}$, can be computed by using a two-stage ϵ-constraint formulation and searching the space whose dimension is one less than the search space associated with enumerating all nondominated solutions by using ε-constraint method [23]. In terms of dimension of the problem, this theorem is similar to Ehrgott and Tenfelde-Podehl's result [17] since they find the nadir point by generating efficient sets of p different MOPs with $(p-1)$ objectives. On the other hand, their result relies on the union of these efficient sets of lesser dimensional MOPs and identifies the nadir point by studying this set. Our result, however, suggests a component-wise identification of the nadir point. This component-wise approach helps us remove some portion of the initial search space by relating the indices that are identified in the payoff table.

By Theorems 1 and 2, with appropriate choice of $\varepsilon \in \mathbb{R}^{p-2}$ the kth component of the nadir point can be obtained by solving the two-stage formulations. To facilitate a search for the appropriate ε values, we can partition the search space into $(p-2)$-dimensional

rectangles. A similar search was implemented in [23] with $(p-1)$-dimensional rectangles to obtain the entire nondominated set. Lower and upper vertices of a rectangle are denoted as $l \in \mathbb{R}^{p-2}$ and $u \in \mathbb{R}^{p-2}$, respectively. With these bounds, a rectangle is defined as $R(l,u) = \{\bar{y} \in \mathbb{R}^{p-2} : l \leq \bar{y} \leq u\}$. For ease of notation, we define $\bar{y} = \bar{f}(x) \in \mathbb{R}^{p-2}$ such that $\bar{f}_j(x) = f_j(x)$ for all $j \in C$, i.e. $\bar{y} = \bar{f}(x)$ is the projection of $f(x) \in \mathbb{R}^p$ onto $\mathbb{R}^{p-2}$ with the mth and kth components removed. Define a rectangle $R(\bar{y}^I, \bar{y}^U)$ whose lower and upper vertices are projections of y^I and y^U onto $\mathbb{R}^{p-2}$, respectively. By definition of y^I and y^U, $y^I_j \leq f_j(x) < y^U_j$ for each $j \in \{1,\ldots,p\}$ and for all $x \in \mathcal{X}$. Hence, all efficient solutions are mapped into $R(\bar{y}^I, \bar{y}^U)$.

The idea is to start with a sufficiently large rectangle that contains the projection of the nondominated set, devise a method of refining the rectangles and solve two-stage problems repeatedly. We will now describe how payoff table information can be used to narrow down the search space.

Let $m \in \{1,\ldots,p\}$ be such that $T_{mk} = y^{PT}_k$, i.e. m is the row that generates the kth component of y^{PT}. Let x^*_m denote an efficient solution that maps into the mth row of the payoff table. Define $\varphi = \bar{f}(x^*_m) \in \mathbb{R}^{p-2}$ where $\varphi_j = f_j(x^*_m)$ for each $j \in C$, i.e. φ is the projection of $f(x^*_m) \in \mathbb{R}^p$ onto $\mathbb{R}^{p-2}$.

Our next result essentially states that $R(\varphi, \bar{y}^U)$ can be excluded from the search space. This reduces our search space to a difference of two rectangles. We now argue that our search space can be represented as a union of several rectangles because $R(\bar{y}^I, \bar{y}^U)$ can be divided into several non-overlapping (in the sense that the interiors do not intersect) subrectangles by pivoting on $\varphi \in \mathbb{R}^{p-2}$. Let $C' = \{j \in C : \bar{y}^I_j < \varphi_j\}$. Consider a rectangular bisection process in which a rectangle is subdivided into two subrectangles by means of a hyperplane $H_j = \{\bar{y} \in \mathbb{R}^{p-2} : \bar{y}_j = \varphi_j\}$ where $j \in C'$. When such a process is applied repeatedly, it generates a family of rectangles which are partitions of the original rectangle [20,26]. An element of this family is of the form

$$R^J = \{\bar{y} \in \mathbb{R}^{p-2} : \begin{array}{ll} \bar{y}^I_j \leq \bar{y}_j \leq \varphi_j & j \in J \\ \varphi_j \leq \bar{y}_j \leq \bar{y}^U_j & j \in C \backslash J\} \end{array} \tag{3}$$

where $J \subseteq C'$. Therefore, union of all such rectangles constitute the initial search space, i.e. $R(\bar{y}^I, \bar{y}^U) = \bigcup_{J \subseteq C'} R^J$. The rectangle $R(\varphi, \bar{y}^U)$ is computed when $J = \emptyset$ in (3). Then the initial search space U_R, given by

$$U_R = \bigcup_{\substack{J \subseteq C' \\ J \neq \emptyset}} R^J \tag{4}$$

contains partitions of $R(\bar{y}^I, \bar{y}^U)$ except for $R(\varphi, \bar{y}^U)$.

The result below shows that the nadir point y^N is projected into U_R where $U_R \subseteq R(\bar{y}^I, \bar{y}^U) \subseteq \mathbb{R}^{p-2}$.

Theorem 3 *Let $m \in \{1,\ldots,p\}$ be such that $T_{mk} = y^{PT}_k$. Let $x^*_m \in \mathcal{X}_E$ be such that $f_j(x^*_m) = T_{mj}$ for each $j \in \{1,\ldots,p\}$. Define $\varphi = \bar{f}(x^*_m) \in \mathbb{R}^{p-2}$. For an efficient solution x' where $f_k(x') = y^N_k$, $\bar{f}(x') \in U_R$.*

Proof Let $x' \in \mathcal{X}$ be an efficient solution such that $f_k(x') = y^N_k$. Assume to the contrary that $f(x')$ is projected outside of U_R. By definition of y^I and y^U, $\bar{f}(x) \in R(\bar{y}^I, \bar{y}^U)$ for all $x \in \mathcal{X}$. This implies that $\bar{f}(x') \in R(\varphi, \bar{y}^U)$. Then $\varphi_j = f_j(x^*_m) \leq f_j(x') \leq y^U_j$ for all $j \in C$. Since $\varphi \in R(\bar{y}^I, \varphi) \subseteq U_R$ and $\bar{f}(x') \notin U_R$, there exists $\hat{\jmath} \in C$ such that

$f_j(x^*_m) < f_j(x')$. By definition of ideal and nadir point, $f_m(x^*_m) = y^I_m \leq f_m(x')$ and $f_k(x^*_m) = y^{PT}_k \leq f_k(x') = y^N_k$. Since $f_j(x^*_m) \leq f_j(x')$ for all $j \in C$ with at least one strict inequality, x^*_m dominates x'. This contradicts efficiency of x'. □

By this and previous theorems, the kth component of the nadir point (y^N_k) can be obtained by an exhaustive search of the region U_R. The search space is represented by a union of $(p-2)$-dimensional rectangles. Unless the number of objective functions is less than or equal to 3, the search space is non-convex. For $p = 2$, the search space is a point. This means that y^{PT} is equal to y^N, which is a known fact. When $p = 3$, the search space becomes a single dimensional interval for the kth component of the nadir point.

In the next section, we provide an algorithm that conducts this search for MODO problems. While results presented so far are applicable to any MOP, our search procedure requires $\mathcal{Y}_{ND}$ to be a discrete set.

4 Finding the nadir point for a MODO problem

In this section, we present an algorithm to search the region U_R exhaustively for all nondominated solutions that map into this region. Since $\varepsilon \in \mathbb{R}^{p-2}$, the search is managed over $(p-2)$-dimensional rectangles. During the search, a set of rectangles should be maintained. The ith rectangle is denoted as R_i with its lower and upper vertices $l^i, u^i \in \mathbb{R}^{p-2}$. All rectangles that need to be searched are kept in list L, i.e. $L = \bigcup_{i=1}^{|L|}\{R_i\}$ where $|L|$ represents the cardinality of L.

Initially, we obtain the efficient solution x^*_m by using the payoff table, where $f_k(x^*_m) = y^{PT}_k$ and $f_m(x^*_m) = y^I_m$. Given objective function indices $\{k, m\}$ and x^*_m, the search space U_R can be constructed by using (3) and (4), and L is initialized as U_R.

After the initialization phase, $L = \emptyset$ implies that y^{PT} is the nadir point. If L is a nonempty set, than we need to process the rectangles in an order. For prioritization, we define a volume-related measure associated with rectangle R_i as

$$V_i = \prod_{j \in C} \left(u^i_j - \bar{y}^I_j \right). \tag{5}$$

Note that this is not the volume of R_i itself, but the volume of the rectangle defined by $\bar{y}^I$ as the lower vertex and the upper vertex u^i of R_i. Theoretically, this prioritization rule can be replaced by any other list processing rule.

The technical statement of the proposed algorithm, which we label as Nadir Point Determination Algorithm (NPDA), is as follows.

Algorithm NPDA
Input: MODO problem, i.e. objective functions $f_j(x)$ for $j = \{1, \ldots, p\}$, and feasible set $\mathcal{X}$
Output: Nadir point (y^N) of MODO problem.

Step-0 Obtain the payoff table T, and initialize the nadir point with the payoff table estimate, i.e. $y^N_j = y^{PT}_j$ for $j = 1, \ldots, p$. Set $k = 1$.
Step-1 Initialize the nondominated solutions list, $\mathcal{Y}^k_P$[1] $= \emptyset$. Pick $m \in \{1, \ldots, p\}$ and efficient

[1] $\mathcal{Y}^k_P$ represents the list of nondominated solutions until y^N_k is obtained. At termination, $\mathcal{Y}^k_P$ may not include all nondominated solutions, i.e. $\mathcal{Y}^k_P \subseteq \mathcal{Y}_{ND}$ for each $k \in \{1, \ldots, p\}$.

solution x_m^* such that $f_k(x_m^*) = y_k^{PT}$. Generate the initial rectangle list L by using (3) and (4) where $\varphi_j = f_j(x_m^*)$ for each $j \in C$.
Step-2 If L is empty, go to Step-4. Otherwise, pick a rectangle $R_i(l^i, u^i)$ with highest V_i from the list L. Solve the first stage formulation with the upper vertex of R_i, i.e. $P_{k,m}(u^i)$.
Step-3 If $P_{k,m}(u^i)$ is feasible, than solve $Q_{k,m}(u^i)$. Let x^* denote an optimal solution.

- If $f(x^*) \notin \mathcal{Y}_P^k$, $\mathcal{Y}_P^k = \mathcal{Y}_P^k \cup \{f(x^*)\}$. If $f_k(x^*) > y_k^N$, update the k^{th} component of the nadir point, $y_k^N = f_k(x^*)$. Now, apply the rectangular subdivision process for each $R_s \in L$. Pick rectangle R_s from list L. Set $L = L \setminus \{R_s\}$. Define an index set for R_s as $C_s' = \{j \in C : l_j^s < \bar{f}_j(x^*) < u_j^s\}$. Initialize list L' for the rectangular subdivision process, $L' = \{R_s\}$.

For Each $(j \in C_s')$
  Initialize list L'', $L'' = \emptyset$.
  For Each $(R_t \in L')$
    $R_1 = \{\bar{y} \in R_t : \bar{y}_j \leq f_j(x^*)\}$
    $R_2 = \{\bar{y} \in R_t : \bar{y}_j \geq f_j(x^*)\}$
    $L'' = L'' \cup \{R_1\} \cup \{R_2\}$
  end
  $L' = L''$
end
$L = L \cup L'$. Remove rectangles that lie in $R(\bar{f}(x^*), u^i)$.

- Else (If $f(x^*) \in \mathcal{Y}_P^k$), remove rectangles that lie in $R(\bar{f}(x^*), u^i)$.

Else (If $P_{k,m}(u^i)$ is infeasible), remove rectangles that lie in $R(\bar{y}^I, u^i)$.

Go to Step-2.

Step-4 $k = k + 1$. If $k \leq p$, go to Step-1. Otherwise, return the nadir point y^N and stop.

NPDA starts with obtaining the payoff table and an efficient solution that corresponds to each component of y^{PT}. Then initial rectangle list L is generated. In each iteration, the algorithm picks a rectangle R_i with the largest V_i value. With the upper vertex of R_i, $u^i \in \mathbb{R}^{p-2}$, the two-stage programs are solved. Until termination, the kth component of the nadir point is determined. This procedure is repeated for each $k \in \{1, \ldots, p\}$ to obtain the nadir point y^N.

While solving the two-stage formulations with $u^i \in \mathbb{R}^{p-2}$, three cases can occur. First a new nondominated solution may be obtained, and let x^* be an associated efficient solution. Initially, $f(x^*)$ is inserted into nondominated solutions list $\mathcal{Y}_P^k$. Then, among all rectangles in L, if $\bar{f}_j(x^*)$ is in between the bounds of rectangle R_s for the jth axis, then R_s is split into two rectangles along the jth axis where $j \in C$.

After updating the list, some of the rectangles can be removed because the region between the upper vertex of the rectangle R_i and $\bar{f}(x)$ does not need to be searched (see Lemma 1). In this case, all rectangles that lie in $R(\bar{f}(x^*), u^i)$ are removed, i.e. for any $R_s \in L$ such that $R_s \subseteq R(\bar{f}(x^*), u^i)$, $L = L \setminus \{R_s\}$.

Lemma 1 *Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$, and let x^* be an optimal solution to two-stage programs $P_{k,m}(u^i)$ and $Q_{k,m}(u^i)$ where $u^i \in \mathbb{R}^{p-2}$ is the upper vertex of a rectangle. Then, there is no nondominated solution that is projected into $R(\bar{f}(x^*), u^i)$, other than $f(x^*)$.*

Proof Assume to the contrary that there exists an efficient solution $x' \in \mathcal{X}$ that is mapped into the rectangle $R(\bar{f}(x^*), u^i)$. Then, $\bar{f}_j(x^*) \leq \bar{f}_j(x') \leq u^i_j$ for each $j \in C$, and $f_k(x') \leq y^U_k$. This implies that x' is feasible to $P_{k,m}(u^i)$, and we know that x^* is optimal to $P_{k,m}(u^i)$. Then, $f_m(x^*) \leq f_m(x')$. Unless $f_j(x') = f_j(x^*)$ for all $j = 1, \ldots, p$, x^* dominates x'. Therefore, there exists no nondominated solution that is mapped into the rectangle $R(\bar{f}(x^*), u^i)$ other than $f(x^*)$. □

The second possible case after solving a set of two-stage programs in NPDA is that an optimal solution x^* is obtained, however $f(x^*)$ has already been found at an earlier iteration, i.e. $f(x^*) \in \mathcal{Y}^k_P$. The reason for this situation is that different $\varepsilon \in \mathbb{R}^{p-2}$ values may return the same nondominated solution. In this case, Lemma 1 still applies, and rectangles that lie in between $\bar{f}(x^*)$ and current rectangle's upper vertex u^i are removed.

The third possible case is the infeasibility of the two-stage programs. Given upper vertex u^i for the rectangle, if $P_{k,m}(u^i)$ is feasible, then $Q_{k,m}(u^i)$ is also feasible. Therefore, infeasibility can be observed only in the first stage problem. If the first stage is infeasible for the rectangle's upper vertex u^i, every $\varepsilon \in R(\bar{y}^I, u^i)$ would lead to an infeasible $P_{k,m}(\varepsilon)$ formulation. Hence, all rectangles inside $R(\bar{y}^I, u^i)$ are removed. Lemma 2 addresses the infeasibility case.

Lemma 2 *Let $k, m \in \{1, \ldots, p\}$ and $k \neq m$, if $P_{k,m}(u^i)$ is infeasible, then there is no nondominated solution that is projected into the rectangle $R(\bar{y}^I, u^i)$.*

Proof Assume to the contrary that there exists an efficient solution $x' \in \mathcal{X}$ such that $f_j(x') \leq u^i_j$ for each $j \in C$, and $y_k(x') \leq y^U_k$. This implies that x' is feasible to $P_{k,m}(u^i)$ which contradicts that $P_{k,m}(u^i)$ has no feasible solution. Therefore there exists no nondominated solution that is mapped into $R(\bar{y}^I, u^i)$. □

We have shown that in each iteration the algorithm removes at least one rectangle from list L, and no other nondominated solution projects into removed rectangles. Similar proofs are given in [23]. Theorems 4 and 5 will show completeness and finiteness of the algorithm after an illustrative example on building and managing rectangles.

Example Consider a four-objective problem ($p = 4$) where the third component ($k = 3$) of the nadir point, y^N_3, is being computed. Let the third component of the payoff table estimate be obtained with $x^*_4 \in \mathcal{X}_E$, i.e. $m = 4$, $f_3(x^*_4) = y^{PT}_3$ and $f_4(x^*_4) = y^I_4$. Since $k = 3$ and $m = 4$, the index set C is equal to $\{1, 2\}$. The projection of $f(x^*_4)$ onto $\bar{y}_1 - \bar{y}_2$ plane is denoted as φ, and suppose that $\bar{y}^I_j < \varphi_j$ for all $j \in C$. Hence, $C' = \{1, 2\}$. The initial search space that contains 3 rectangles is $U_R = R((\bar{y}^I_1, \varphi_2), (\varphi_1, \bar{y}^U_2)) \cup R((\varphi_1, \bar{y}^I_2), (\bar{y}^U_1, \varphi_2)) \cup R((\bar{y}^I_1, \bar{y}^I_2), (\varphi_1, \varphi_2))$ as shown in Fig. 1. In this figure shaded region shows the removed space, which is $R((\varphi_1, \varphi_2), (\bar{y}^U_1, \bar{y}^U_2))$.

After the initialization phase, the procedure continues by picking $R((\bar{y}^I_1, \varphi_2), (\varphi_1, \bar{y}^U_2))$. Two-stage formulation is solved for the upper vertex of this rectangle, and the projection of the resulting nondominated solution is $\bar{y}^1$. Rectangular subdivision process pivoting on $\bar{y}^1$ is shown in Fig. 2. After the rectangular subdivision process, rectangles that lie in $R((\bar{y}^1_1, \bar{y}^1_2), (\varphi_1, \bar{y}^U_2))$ can be removed. Thus two rectangles, $R((\bar{y}^1_1, \varphi_2), (\varphi_1, \bar{y}^U_2))$ and $R((\bar{y}^1_1, \bar{y}^1_2), (\varphi_1, \varphi_2))$, are removed.

Theorem 4 *NPDA finds the nadir point y^N.*

Proof Two-stage mathematical programs either find an efficient solution (see Theorem 1) or they are infeasible for right-hand-side vector u^i. Therefore, all points in $\mathcal{Y}^k_P$ for each $k \in \{1, \ldots, p\}$, that are obtained during the search are nondominated. In NPDA, the search

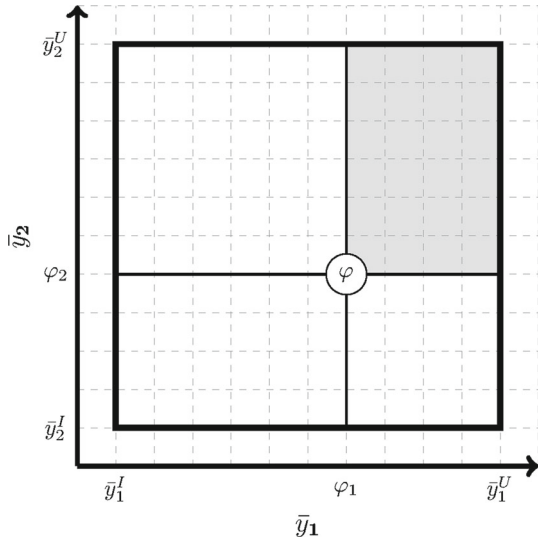


Fig. 1 Initial search space and the portion removed by using payoff estimate

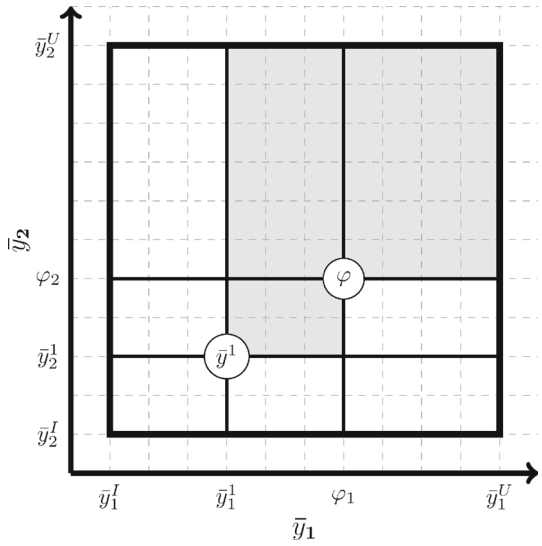


Fig. 2 Rectangular subdivision process pivoting on $\bar{y}^1$ and removed rectangles

is initialized with a set of rectangles that forms the search space U_R. By Theorem 3, an efficient solution x^* that satisfies $f_k(x^*) = y_k^N$ is mapped into this region. In NPDA, this space is searched exhaustively, and in each iteration some part of the initial rectangle is removed. From Lemmas 1 and 2, removed regions do not include projection of a nondominated solution. By Theorem 2, there exists $\varepsilon \in \mathbb{R}^{p-2}$ such that x^* is optimal to a two-stage formulation whenever $f_k(x^*) = y_k^N$. Since NPDA considers each $k \in \{1, \ldots, p\}$, at termination NPDA obtains the nadir point y^N. □

We also need to argue the finiteness of NPDA. In Theorem 5, we show that the proposed algorithm terminates in a finite number of iterations.

Theorem 5 *NPDA is finite.*

Proof By assumption, the nondominated set $\mathcal{Y}_{ND}$ includes a finite number of elements. Since the list of nondominated solutions until y_k^N is obtained ($\mathcal{Y}_P^k$) is a subset of $\mathcal{Y}_{ND}$, $\mathcal{Y}_P^k$ also has a finite number of elements, $|\mathcal{Y}_P^k|$. In the worst case, each axis in $\mathbb{R}^{p-2}$ is divided into $(|\mathcal{Y}_P^k|+1)$ intervals. Since the search space is $(p-2)$-dimensional, $|L| \leq (|\mathcal{Y}_P^k|+1)^{p-2}$ where $|L|$ denotes the size of the rectangle list. This means that the number of rectangles is finite. Besides, in each iteration at least one rectangle is removed from L. This implies that L will be empty in a finite number of iterations. Hence, NPDA is finite. □

The algorithm terminates when the list of rectangles is empty. Theorems 4 and 5 imply that NPDA finds the nadir point y^N in a finite number of iterations.

5 Computational results

NPDA, Ehrgott and Tenfelde-Podehl's algorithm (ETPA), and Jorge's algorithm (JA) are tested on various MODO problem instances. Ehrgott and Tenfelde-Podehl showed that it suffices to generate efficient sets of p different MODOs with $(p-1)$ objective functions to compute the nadir point. Hence, their method requires an algorithm to enumerate all efficient solutions for MODO with $(p-1)$ objective functions. Therefore, we modify Kirlik and Sayın's nondominated set enumeration algorithm so as to generate the entire efficient set by finding all alternative optimal solutions for each subproblem. We choose Kirlik and Sayın's algorithm, because it performs well in terms of solution time [23], and utilizes a search structure similar to the one presented here, making a comparison more straight forward in terms of highlighting the additional benefits of NPDA. In the computational results, ETPA [17] with modified Kirlik & Sayın's algorithm [23] is denoted as ETPA+. The proposed method is also compared with a recent solution algorithm for the problem of optimizing a linear function over the integer efficient set, which is presented by Jorge [22]. JA iteratively cuts previously obtained nondominated solutions until the kth component of the nadir point is obtained. Hence, the solution algorithm adds p binary variables and $p+1$ constraints for each nondominated solution to the model.

All algorithms are implemented in C++. Subproblems are solved by using IBM CPLEX 12.4 [12]. All tests were conducted on a shared cluster with Intel Xeon 2.3 GHz CPU and 4 GB memory limit with Linux operating system. These algorithms are tested on the multiobjective knapsack problem, multiobjective assignment problem, and multiobjective integer linear programming problem instances. Different problem categories are generated based on problem size, and 10 instances are generated randomly for each problem category. The results of each category are presented in two rows. In the first row, for each category, the average over 10 instances is given. The numbers in the second row correspond to the minimum and maximum for each category, respectively. Computation of each instance is interrupted after 100,000 CPU seconds. A blank cell in a table indicates that none of the 10 instances could be completed within the time limit.

For NPDA, we report total size of the list $\mathcal{Y}_P^k$ over all components of the nadir point, i.e. $\sum_{k=1}^{p} |\mathcal{Y}_P^k|$ is reported. Number of models solved and CPU time statistics are given for all algorithms.

5.1 The multiobjective knapsack problem

The multiobjective knapsack problem consists of n objects, rth object in the knapsack has a positive integer weight w_r and p non-negative integer profits v_r. The knapsack has a positive

Table 1 Comparing the solution methods on the multiobjective knapsack problem with $p = 3$

n	ETPA+		JA		NPDA		
	#Models	CPU time (s)	#Models	CPU time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU time (s)
10	15.9	0.1	9.9	0.2	9.9	12.9	0.1
	[10, 25]	[0.0, 0.1]	[8, 12]	[0.1, 0.2]	[4, 19]	[7, 22]	[0.0, 0.1]
20	35.9	0.4	23.1	1.4	30.0	33.0	0.2
	[28, 44]	[0.3, 0.4]	[18, 32]	[0.9, 2.8]	[22, 38]	[25, 41]	[0.1, 0.3]
30	61.0	1.0	39.4	8.4	55.1	58.1	0.5
	[46, 80]	[0.7, 1.4]	[22, 72]	[2.3, 28.7]	[40, 74]	[43, 77]	[0.3, 0.7]
40	104.7	3.4	70.8	57.6	99.0	102.0	1.3
	[77, 169]	[1.9, 7.5]	[42, 130]	[9.2, 253.5]	[71, 163]	[74, 166]	[0.8, 2.6]
50	138.7	6.4	89.2	132.5	133.1	136.1	2.3
	[82, 201]	[2.2, 15.7]	[42, 173]	[12.1, 625.5]	[76, 196]	[79, 199]	[0.9, 5.2]
60	188.6	12.9	154.2	830.8	183.3	186.3	3.8
	[153, 236]	[7.4, 23.8]	[110, 218]	[136.3, 2,411.7]	[147, 232]	[150, 235]	[2.4, 6.0]
70	241.1	27.4	194.7	1782.2	237.6	240.6	6.5
	[194, 295]	[16.0, 34.9]	[119, 333]	[176.7, 5,370.0]	[190, 302]	[193, 305]	[4.3, 8.2]
80	329.2	51.5	238.7	3993.1	327.2	330.2	11.9
	[215, 431]	[27.1, 76.9]	[135, 378]	[230.7, 16,912.9]	[212, 427]	[215, 430]	[7.5, 16.7]
90	362.5	62.0	298.2	7976.8	359.1	362.1	13.9
	[258, 457]	[32.0, 92.0]	[137, 472]	[178.4, 24,084.0]	[253, 456]	[256, 459]	[7.4, 21.1]
100	470.5	103.7	405.3	**{3}**[a] 24,052.6	468.8	471.8	22.8
	[330, 651]	[59.3, 181.7]	[205, 613]	[1,205.2, 63,326.5]	[326, 658]	[329, 661]	[12.9, 39.8]

[a] Number of instances that could not be solved in 100,000 CPU seconds

integer capacity W. Decision variable x_r denotes whether item r is selected for the knapsack or not.

$$\begin{aligned} (\textbf{MOKP}) \quad \max \ & \sum_{r=1}^{n} v_r^j x_r \quad j = 1, \ldots, p \\ \text{s.t.} \ & \sum_{r=1}^{n} w_r x_r \leq W \\ & x_r \in \{0, 1\} \quad r = 1, \ldots, n \end{aligned}$$

The multiobjective knapsack problem instances are generated for $p = 3$, 4 and 5 cases, and v_r^j and w_r are random integers drawn from the interval [1, 1,000] where $j \in \{1, \ldots, p\}$ and $r \in \{1, \ldots, n\}$. The capacity of the knapsack is calculated as $W = \lceil 0.5 \sum_{r=1}^{n} w_r \rceil$. For $p = 3$ case, we have generated 10 categories with a size varying from 10 to 100 with increments of 10. The solution statistics are given in Table 1.

As seen in Table 1, NPDA outperforms both algorithms in terms of CPU time results. We observe that JA cannot solve all instances in the given time limit. When we compare NPDA with ETPA+, the number of solved models in each problem category for both methods are almost equal, but ETPA+ enumerates all alternative solutions for the given subproblem. In all instances, NPDA is able to obtain the nadir point in a shorter period of time.

For $p = 4$ and $p = 5$ cases, we have generated 4 and 3 categories with a size varying from 10 to 40 and 10 to 30 with increments of 10, respectively. The solution statistics for both cases are given in Table 2.

For $p = 4$ case, NPDA still outperforms other two methods. In these instances, the solution time of NPDA and ETPA+ are in reasonable limits. This is not the case for JA. When we switch to $p = 5$ test instances, JA outperforms NPDA with the difference being important for $n = 30$. However, NPDA still outperforms ETPA+ for both $p = 4$ and $p = 5$ cases.

5.2 The multiobjective assignment problem

The assignment problem aims to obtain optimal assignments between a set of agents $r \in \{1, \ldots, n\}$ and a set of tasks $l \in \{1 \ldots, n\}$ where each assignment has a non-negative cost c_{rl}. The multiobjective assignment problem is formulated as follows.

$$
\begin{aligned}
(MOAP) \quad \min \ & \sum_{r=1}^{n} \sum_{l=1}^{n} c_{rl}^{j} x_{rl} \quad j = 1, \ldots, p \\
\text{s.t.} \ & \sum_{l=1}^{n} x_{rl} = 1 \quad r = 1, \ldots, n \\
& \sum_{r=1}^{n} x_{rl} = 1 \quad l = 1, \ldots, n \\
& x_{rl} \in \{0, 1\} \quad r = 1, \ldots, n; \ l = 1, \ldots, n
\end{aligned}
$$

Test problem instances for the multiobjective assignment problem are formed in sizes varying from 5 to 50 with increments of 5 and $p = 3$. The objective function coefficients are generated randomly in the interval [1, 20], and all are integers. The results are given in Table 3.

As seen in Table 3, NPDA computes the nadir point faster than ETPA+, and the performance difference is increasing with the increase in problem size. Another observation is that CPU time difference between NPDA and ETPA+ is larger in assignment problem instances compared to knapsack problem instances with $p = 3$. JA spends too much time to find the nadir on multiobjective assignment problems, and the algorithm cannot solve more than half of the instances in 100,000 CPU seconds. The assignment problem instances have larger number of nondominated solutions compared to knapsack problem instances (see [23] for the statistics). Hence, JA requires to add many binary variables and constraints to the model, and the model becomes unsolvable even if the problem size is $n = 25$.

We note that similar multiobjective knapsack and assignment problem instances were generated and their nondominated sets were enumerated in [23]. While the average number of nondominated solutions for the largest three-objective knapsack ($p = 3$ and $n = 100$) and assignment ($p = 3$ and $n = 50$) problem instances is 5,849.0 and 24,916.8, NPDA has $\sum_{k=1}^{p} |\mathcal{Y}_P^k|$ equal to 470 and 700.8, respectively. This implies that NPDA enumerates a small portion of $\mathcal{Y}_{ND}$ in the process.

5.3 The multiobjective integer linear problem

We test the algorithms on general multiobjective integer linear programming (MOILP) problem instances. Here, m and n represent the number of constraints and number of variables, respectively, and x is the decision vector of the problem. Given coefficients of the

Table 2 Comparing the solution methods on the multiobjective knapsack problem with $p = 4$ and $p = 5$

	n	ETPA+		JA		NPDA		
		#Models	CPU time (s)	#Models	CPU time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU time (s)
$p = 4$	10	59.8	0.3	14.7	0.2	26.0	55.8	0.2
		[32, 94]	[0.1, 0.6]	[12, 21]	[0.1, 0.4]	[12, 43]	[28, 90]	[0.1, 0.4]
	20	407.9	7.4	40.5	7.1	200.7	404.4	4.6
		[90, 729]	[0.9, 14.1]	[17, 58]	[0.7, 20.9]	[41, 361]	[86, 723]	[0.7, 9.0]
	30	942.0	27.9	93.3	104.2	469.3	938.5	15.2
		[554, 1,526]	[11.8, 51.5]	[57, 134]	[19.5, 236.8]	[274, 762]	[549, 1,522]	[6.4, 27.7]
	40	2,898.4	230.1	179.7	2,641.5	1,459.5	2,903.5	98.2
		[1,368, 6,055]	[55.0, 615.8]	[106, 287]	[105.9, 10,259.5]	[689, 3,066]	[1,371, 6,096]	[24.5, 259.3]
$p = 5$	10	230.1	1.3	21.7	0.4	56.4	224.8	0.9
		[86, 454]	[0.2, 3.1]	[13, 25]	[0.2, 0.6]	[21, 108]	[81, 446]	[0.1, 2.3]
	20	1,952.7	75.5	56.0	17.3	424.8	1,948.2	60.0
		[862, 4,770]	[13.1, 326.9]	[40, 73]	[3.8, 67.1]	[210, 958]	[857, 4,769]	[9.5, 284.0]
	30	12,630.0	20,064.8	164.4	2,353.6	2,423.5	12,648.9	18,593.9
		[1,617, 23,718]	[44.8, 74,862.9]	[75, 233]	[60.4, 9,333.0]	[369, 4,407]	[1,614, 23,754]	[26.6, 71,104.3]

Table 3 Comparing the solution methods on the multiobjective assignment problem with $p = 3$

n	ETPA+		JA		NPDA		
	#Models	CPU time (s)	#Models	CPU time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU time (s)
5	16.4	0.1	45.0	17.0	10.8	13.8	0.1
	[11, 25]	[0.0, 0.1]	[30, 61]	[5.9, 26.1]	[5, 19]	[8, 22]	[0.0, 0.1]
10	60.2	1.2	145.0	491.4	62.5	65.5	0.9
	[38, 79]	[0.7, 1.8]	[101, 188]	[199.6, 945.9]	[41, 80]	[44, 83]	[0.6, 1.1]
15	103.9	5.0	376.5	7231.3	118.1	121.1	2.8
	[95, 120]	[4.5, 5.9]	[212, 505]	[1,685.2, 12,507.4]	[104, 140]	[107, 143]	[2.3, 3.6]
20	169.0	15.2	376.5	7,315.1	202.9	205.9	7.2
	[129, 206]	[10.8, 19.0]	[212, 505]	[1,716.8, 13,093.7]	[155, 255]	[158, 258]	[5.6, 9.4]
25	199.3	33.0	644.3	**{4}**[a] 45,850.1	248.4	251.4	13.2
	[164, 231]	[26.0, 38.6]	[458, 806]	[20,841.2, 80,308.3]	[195, 291]	[198, 294]	[10.3, 15.6]
30	268.9	67.5			370.4	373.4	26.2
	[241, 311]	[59.8, 80.0]			[330, 417]	[333, 420]	[23.4, 29.6]
35	312.3	124.5			438.5	441.5	40.5
	[282, 346]	[100.7, 139.4]			[386, 524]	[389, 527]	[33.7, 49.3]
40	359.5	205.8			536.1	539.1	51.6
	[329, 404]	[189.8, 221.9]			[462, 616]	[465, 619]	[44.8, 56.6]
45	390.0	321.4			602.9	605.9	74.1
	[354, 421]	[289.8, 349.8]			[516, 713]	[519, 716]	[65.3, 91.1]
50	427.9	507.8			700.8	703.8	105.1
	[399, 442]	[458.1, 541.9]			[649, 783]	[652, 786]	[86.9, 116.3]

[a] Number of instances that could not be solved in 100,000 CPU seconds

objective functions c_l^j, the technical coefficients a_{rl}, and right-hand side values b_r where $r \in \{1, \ldots, m\}, l \in \{1, \ldots, n\}$, and $j \in \{1, \ldots, p\}$, MOILP problem is defined as follows.

$$
\begin{aligned}
(MOILP) \quad \max \ & \sum_{l=1}^{n} c_l^j x_l \quad j = 1, \ldots, p \\
\text{s.t.} \ & \sum_{l=1}^{n} a_{rl} x_l \leq b_r \quad r = 1, \ldots, m \\
& x_l \geq 0 \text{ and integer} \quad l = 1, \ldots, n.
\end{aligned}
$$

We consider MOILP problems with $p = 3, 4$ and 5, $m = 5, 10, \ldots, 50$ and $n = 2m$ for each m. The parameters of the model are randomly generated integer numbers with ranges similar to used in [2]. The coefficients of the objective functions (c_l^j) are generated in the ranges $[-100, -1]$ and $[0, 100]$ with probability 0.2 and 0.8, respectively. The technical coefficients (a_{rl}) are generated in the ranges $[-100, -1]$ with probability 0.1, $[1, 100]$ with probability 0.8, and $a_{rl} = 0$ with probability 0.1. Finally, right-hand side value (b_r) of each constraint is also generated randomly in the range of 100 and $\sum_{l=1}^{n} a_{rl}$. According to this scheme, it is possible for a generated MOILP instance to have an unbounded efficient set.

Table 4 Comparing the solution methods on the MOILP problems with $p = 3$

$m \times n$	ETPA+		JA		NPDA		
	#Models	CPU time (s)	#Models	CPU time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU time (s)
5 × 10	30.7	0.4	11.5	0.2	25.2	28.2	0.3
	[8, 83]	[0.1, 1.1]	[4, 16]	[0.0, 0.4]	[2, 80]	[5, 83]	[0.0, 0.7]
10 × 20	30.9	1.2	16.7	0.8	25.1	28.1	0.6
	[15, 51]	[0.3, 2.9]	[8, 35]	[0.2, 2.1]	[9, 46]	[12, 49]	[0.2, 1.1]
15 × 30	47.5	11.4	26.3	8.4	42.4	45.4	4.9
	[26, 79]	[1.5, 32.2]	[17, 35]	[1.8, 18.1]	[20, 74]	[23, 77]	[0.6, 12.4]
20 × 40	63.9	47.8	30.9	46.7	59.3	62.3	21.8
	[28, 125]	[8.0, 153.2]	[16, 57]	[5.1, 307.6]	[24, 121]	[27, 124]	[3.5, 74.3]
25 × 50	71.6	130.1	45.5	142.8	66.6	69.6	71.8
	[37, 122]	[5.8, 414.6]	[22, 74]	[6.0, 660.7]	[31, 117]	[34, 120]	[3.5, 253.0]
30×60	71.5	257.9	53.9	1143.7	67.1	70.1	169.7
	[46, 130]	[37.6, 1,317.2]	[22, 141]	[10.3, 9917.0]	[41, 127]	[44, 130]	[16.0, 1,058.3]
35 × 70	81.9	495.4	62.1	1200.7	78.1	81.1	323.6
	[27, 140]	[22.3, 1,130.2]	[15, 158]	[10.5, 6,321.4]	[22, 139]	[25, 142]	[7.9, 803.5]
40 × 80	76.4	568.1	49.4	1013.4	71.4	74.4	405.5
	[47, 146]	[59.9, 2,317.6]	[26, 95]	[64.5, 3,213.0]	[41, 142]	[44, 145]	[18.5, 1,803.7]
45 × 90	101.6	3,589.5	67.0	9,225.7	98.0	101.0	2537.1
	[40, 190]	[112.1, 16,020.1]	[27, 147]	[86.8, 67,455.5]	[36, 190]	[39, 193]	[47.9, 12,227.9]
50 × 100	94.6	1738.5	56.6	4003.6	89.3	92.3	1,187.1
	[69, 138]	[420.8, 39,56.6]	[33, 106]	[454.5, 17,602.4]	[63, 135]	[66, 138]	[235.7, 2,827.3]

Only one such instance was encountered in the $p = 5$ category and was discarded as reported in Table 6.

For $p = 3$ case, we test all methods with up to 50 constraints and 100 decision variables. The results of these tests are given in Table 4.

As seen in Table 4, the performance of NPDA and ETPA+ are closer compared to knapsack and assignment instances. Nevertheless, NPDA performs better than both methods in all instances. Additionally, NPDA solves all instances in an acceptable period of time even if the problem size increases to 50×100. For $p = 4$ case, we test all methods with up to 80 decision variables. The results of these tests are given in Table 5.

For $p = 4$ case, one out of 10 instances in the 40×80 category cannot be solved by NPDA in the given time limit. In this category, JA outperforms NPDA in small-sized instances. However, NPDA determines the nadir points faster than JA in most of the medium-sized instances.

Finally, algorithms are tested on the randomly generated MOILP problem instances with $p = 5$. For this group of test, the instances are generated with up to 40 decision variables. The results of these tests are given in Table 6. JA is able to solve all instances for $p = 5$ case. With higher number of objective functions, Jorge's algorithm works better than NPDA; however, we have to note that these are small problems in terms of number of variables and constraints. Since NPDA searches the space through $(p - 2)$-dimensional rectangles,

Table 5 Comparing the solution methods on the MOILP problems with $p = 4$

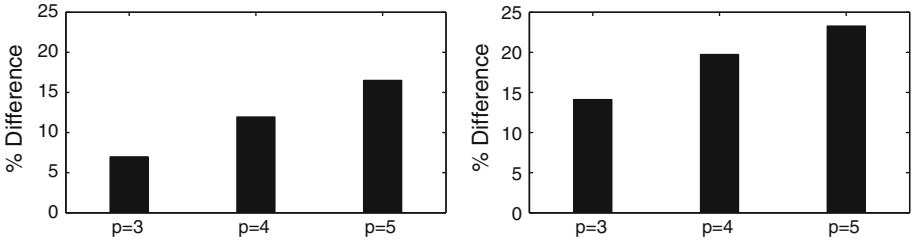
$m \times n$	ETPA+		JA		NPDA		
	#Models	CPU time (s)	#Models	CPU time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU time (s)
5×10	318.8	4.9	18.1	0.7	157.4	314.1	3.8
	[52, 899]	[0.5, 14.3]	[8, 40]	[0.1, 3.1]	[22, 452]	[48, 895]	[0.4, 10.4]
10×20	635.4	33.0	37.1	11.6	320.7	633.4	20.2
	[190, 1,016]	[7.0, 81.3]	[17, 53]	[0.8, 40.4]	[92, 513]	[186, 1,012]	[4.6, 46.2]
15×30	1,124.0	500.9	58.4	324.2	580.0	1,131.2	233.3
	[162, 3,882]	[7.2, 2,539.9]	[24, 164]	[2.4, 2,593.0]	[80, 2,037]	[160, 3,972]	[4.4, 1,074.8]
20×40	1,152.1	1,175.2	60.5	374.6	592.2	1,151.0	635.5
	[388, 2,827]	[71.0, 6,307.7]	[32, 139]	[23.3, 1,210.3]	[196, 1,474]	[389, 2,803]	[38.0, 3,725.0]
25×50	1,030.0	3,522.3	56.4	3,169.2	527.1	1,033.0	2,200.4
	[212, 3,206]	[45.2, 15,961.0]	[25, 118]	[10.3, 24,968.0]	[103, 1,657]	[209, 3,221]	[28.6, 10,423.8]
30×60	2,907.9	17,114.5	110.9	**{3}**[a] 10,950.3	1,507.1	2,924.7	10,950.6
	[1,801, 6,725]	[2,056.2, 43,766.8]	[67, 180]	[409.6, 34,053.4]	[935, 3,492]	[1,801, 6,781]	[1038.4, 29,720.1]
35×70	2,545.4	20,121.8	76.3	**{3}**[a] 4,173.1	1,313.4	2,556.5	14,355.1
	[1,047, 5,463]	[935.6, 43,698.6]	[45, 125]	[61.8, 15,699.3]	[531, 2,829]	[1,039, 5,496]	[393.2, 33,628.6]
40×80	1,768.2	**{2}**[a] 19,503.5	100.6	**{1}**[a] 14,654.3	1,124.3	2,191.8	**{1}**[a] 20,196.6
	[1,060, 2,786]	[8,613.4, 42,920.2]	[55, 189]	[762.7, 63,661.2]	[539, 2,856]	[1,061, 5,510]	[3,907.7, 88,401.0]

[a] Number of instances that could not be solved in 100,000 CPU seconds

Table 6 Comparing the solution methods on the MOILP problems with $p = 5$

$m \times n$	ETPA+		JA		NPDA		
	#Models	CPU time (s)	#Models	CPU time (s)	$\sum_k \lvert\mathcal{Y}_P^k\rvert$	#Models	CPU time (s)
{1}[b] 5×10	2,102.6	129.8	25.1	1.6	457.3	2109.8	121.7
	[169, 6,255]	[0.6, 529.4]	[0, 50]	[0.0, 5.7]	[43, 1,294]	[164, 6,413]	[0.7, 548.7]
10×20	8,153.4	3,637.9	51.7	84.2	1,634.6	8,174.6	3,387.3
	[2018, 13,197]	[81.5, 12,294.7]	[17, 96]	[0.6, 561.5]	[426, 2,580]	[2,024, 13,199]	[55.2, 12,753.8]
15×30	6,062.5	**{2}**[a] 1,756.3	99.0	4,778.0	1,254.9	6,078.1	**{2}**[a] 1,246.3
	[2,707, 8,608]	[212.5, 3,720.9]	[34, 377]	[5.5, 45,078.6]	[619, 1,744]	[2,697, 8,613]	[150.9, 2,532.3]
20×40	7,906.4	**{2}**[a] 8,146.3	103.2	6,264.9	1,626.6	7,951.2	**{2}**[a] 4,972.8
	[1,668, 17,950]	[188.5, 32,858.1]	[43, 242]	[16.9, 39,210.2]	[372, 3,694]	[1,653, 18,172]	[118.4, 21,093.9]

[a] Number of instances that could not be solved in 100,000 CPU seconds
[b] Number of unbounded instances



(a) Multiobjective knapsack problem with $p = 3, 4$ and 5.

(b) Multiobjective integer linear programming problemwith $p = 3, 4$ and 5.

Fig. 3 Percentage difference between y^{PT} and y^N

NPDA's complexity is exponential in number of objective functions. On the other hand, the performance of JA is affected by the size of the nondominated set in general. Therefore, JA does not strongly react to number of objective functions as long as the problem size remains relatively small otherwise and the nondominated set does not grow.

In summary, NPDA outperforms ETPA+ in all instances independent of the problem type and size, despite the fact that we utilize a very competitive algorithm to obtain the efficient set in ETPA+. NPDA also outperforms JA except for $p = 5$ cases for which only small size problems can be reported.

5.4 Payoff estimate and nadir point comparison

In this subsection, we wish to explore the quality of payoff estimation in discrete optimization problems. For a given problem type with N instances and p objective functions, we compute

$$\Delta = 100 \times \sum_{i=1}^{N} \sum_{j=1}^{p} \left(\frac{y_{ij}^N - y_{ij}^{PT}}{y_{ij}^N - y_{ij}^I} \right) \Big/ (N \times p). \tag{6}$$

Δ can be interpreted as the normalized distance between a payoff table estimate and a nadir point computed independently for each objective function. We calculate Δ for multiobjective knapsack and MOILP problem instances.

As seen in Fig. 3, for both problem types, the average distance between the payoff table estimate and the nadir point is increasing with the number of objective functions. On average, the payoff table estimate quality is worse in MOILP than in knapsack problems. This might be one of the underlying factors that lead to closer computational performance of NPDA and ETPA+ in MOILP instances.

6 Conclusion

In this paper, we presented a new algorithm to find the nadir point of MODO problems with p objective functions. The proposed algorithm NPDA is based on an exhaustive search of the $(p-2)$-dimensional space. NPDA guarantees to find the nadir point exactly in a finite number of steps. We also compute Ehrgott and Tenfelde-Podehl's nadir point determination approach, and Jorge's solution method for the optimization over integer efficient set problem. All algorithms are tested on various types of discrete optimization problems with different sizes. In computational results, we see that NPDA outperforms Ehrgott and Tenfelde-Podehl's

algorithm in all test instances by varying margins. NPDA also outperforms Jorge's algorithm in three and four-objective optimization problems. When $p = 5$, Jorge's algorithm displays better performance over problem sizes that can be solved within the enforced time limit.

Acknowledgments This work is supported by TUBITAK (Scientific & Technical Research Council of Turkey), Project No. 112M217. We thank anonymous reviewers whose comments improved the presentation of the paper.

References

1. Abbas, M., Chaabane, D.: Optimizing a linear function over an integer efficient set. Eur. J. Oper. Res. **174**(2), 1140–1161 (2006)
2. Alves, M.J., Costa, J.P.: An exact method for computing the nadir values in multiple objective linear programming. Eur. J. Oper. Res. **198**(2), 637–646 (2009)
3. An, L.T.H., Tao, P.D., Muu, L.D.: Numerical solution for optimization over the efficient set by dc optimization algorithms. Oper. Res. Lett. **19**(3), 117–128 (1996)
4. Ben-Tal, A.: Characterization of pareto and lexicographic optimal solutions. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making Theory and Application. Lecture Notes in Economics and Mathematical Systems, vol. 177, pp. 1–11. Springer, Berlin (1980)
5. Benson, H.P.: Optimization over the efficient set. J. Math. Anal. Appl. **98**(2), 562–580 (1984)
6. Benson, H.P.: An all-linear programming relaxation algorithm for optimizing over the efficient set. J. Glob. Optim. **1**(1), 83–104 (1991)
7. Benson, H.P.: A finite, nonadjacent extreme-point search algorithm for optimization over the efficient set. J. Optim. Theory Appl. **73**(1), 47–64 (1992)
8. Benson, H.P.: A bisection-extreme point search algorithm for optimizing over the efficient set in the linear dependence case. J. Glob. Optim. **3**(1), 95–111 (1993)
9. Benson, H.P., Lee, D., McClure, J.P.: Global optimization in practice: an application to interactive multiple objective linear programming. J. Glob. Optim. **12**(4), 353–372 (1998)
10. Bolintineanu, S.: Minimization of a quasi-concave function over an efficient set. Math. Program. **61**(1), 89–110 (1993)
11. Bonnel, H., Kaya, C.Y.: Optimization over the efficient set of multi-objective convex optimal control problems. J. Optim. Theory Appl. **147**(1), 93–112 (2010)
12. Cplex: IBM ILOG Cplex Optimization Studio 12.4 (2012)
13. Dauer, J.P.: Optimization over the efficient set using an active constraint approach. Math. Methods Oper. Res. **35**(3), 185–195 (1991)
14. Dauer, J.P., Fosnaugh, T.A.: Optimization over the efficient set. J. Glob. Optim. **7**(3), 261–277 (1995)
15. Ecker, J.G., Song, J.H.: Optimizing a linear function over an efficient set. J. Optim. Theory Appl. **83**(3), 541–563 (1994)
16. Ehrgott, M.: Multicriteria Optimization. Springer, Berlin (2005)
17. Ehrgott, M., Tenfelde-Podehl, D.: Computation of ideal and nadir values and implications for their use in MCDM methods. Eur. J. Oper. Res. **151**(1), 119–139 (2003)
18. Granat, J., Guerriero, F.: The interactive analysis of the multicriteria shortest path problem by the reference point method. Eur. J. Oper. Res. **151**(1), 103–118 (2003)
19. Haimes, Y.Y., Lasdon, L.S., Wismer, D.A.: On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Trans. Syst. Man Cybern. **1**(3), 296–297 (1971)
20. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, New York (1995)
21. Isermann, H., Steuer, R.E.: Computational experience concerning payoff tables and minimum criterion values over the efficient set. Eur. J. Oper. Res. **33**(1), 91–97 (1988)
22. Jorge, J.M.: An algorithm for optimizing a linear function over an integer efficient set. Eur. J. Oper. Res. **195**(1), 98–103 (2009)
23. Kirlik, G., Sayın, S.: A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. Eur. J. Oper. Res. **232**(3), 479–488 (2014)
24. Köksalan, M.M., Sagala, P.N.S.: Interactive approaches for discrete alternative multiple criteria decision making with monotone utility functions. Manag. Sci. **41**(7), 1158–1171 (1995)
25. Laumanns, M., Thiele, L., Zitzler, E.: An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. Eur. J. Oper. Res. **169**(3), 932–942 (2006)

26. Le Thi Hoai, A., Tao, P.D.: Solving a class of linearly constrained indefinite quadratic problems by DC algorithms. J. Glob. Optim. **11**(3), 253–285 (1997)
27. Miettinen, K., Eskelinen, P., Ruiz, F., Luque, M.: NAUTILUS method: an interactive technique in multi-objective optimization based on the nadir point. Eur. J. Oper. Res. **206**(2), 426–434 (2010)
28. Özlen, M., Azizoğlu, M.: Multi-objective integer programming: a general approach for generating all non-dominated solutions. Eur. J. Oper. Res. **199**(1), 25–35 (2009)
29. Reeves, G.R., Reid, R.C.: Minimum values over the efficient set in multiple objective decision making. Eur. J. Oper. Res. **36**(3), 334–338 (1988)
30. Sayın, S.: Optimizing over the efficient set using a top-down search of faces. Oper. Res. **48**(1), 65–72 (2000)
31. Yamamoto, Y.: Optimization over the efficient set: overview. J. Glob. Optim. **22**(1), 285–317 (2002)