Boundary Decomposition for Finding Nadir Objective Vector in Multi-Objective Discrete Optimization

Ruihao Zheng, Zhenkun Wang*

School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, China
12132686@mail.sustech.edu.cn, wangzhenkun90@gmail.com

Abstract

The exact nadir objective vector of a multi-objective discrete optimization problem (MODOP) is crucial for decision-making but remains challenging to find. Existing methods for tackling this issue have limitations in theoretical guarantees or high computational costs. This paper applies boundary decomposition to the MODOP and proposes an exact algorithm called BDNC. BDNC is designed to address a bilevel optimization problem for each objective with finite-time convergence guarantees. The lower-level optimization problem, termed the boundary subproblem, is a scalarization of the MODOP. It can be solved using any suitable single-objective exact solver. According to the theoretical foundations of boundary decomposition, some specific settings of the boundary subproblem can ensure alignment with the nadir objective vector under mild conditions. The upper-level optimization problem evaluates a potential setting using the optimal solution to the lower-level one. It employs our proposed novel pruning method to efficiently identify the specific settings. Moreover, BDNC can leverage a trade-off provided by the decision-makers, potentially facilitating the decision-making process. Experiments on various MODOPs demonstrate that BDNC exhibits superior and reliable performance in terms of runtime compared to existing exact methods.

Introduction

Multi-objective optimization involves an optimization scenario where several conflicting objectives must be considered simultaneously, often occurring in real-world applications. There is no single optimal solution to a multi-objective optimization problem (MOP). Instead, there is a set of Pareto-optimal solutions, representing different trade-offs between these objectives. The component-wise worst objective function values of Pareto-optimal solutions define the nadir objective vector. Obtaining the nadir objective vector is crucial for solving MOPs in various applications, such as engineering design (Weerasuriya et al. 2021; Mena et al. 2023) and multi-objective learning (Momma, Dong, and Liu 2022; Zhang et al. 2023). Specifically, the nadir objective vector provides a promising region in the objective space. This serves as a prerequisite for decision-makers to apply preference-based algorithms (*e.g.*, interactive algorithms) (Branke et al. 2008; Wang, Olhofer, and Jin 2017; Mena et al. 2023). Moreover, the nadir objective vector is frequently utilized to guide the search of algorithms in both continuous and discrete optimization (Vodopija, Tušar, and Filipič 2024; Mesquita-Cunha, Figueira, and Barbosa-Póvoa 2023). In addition, the nadir objective vector, along with the ideal objective vector, is widely used for the normalization of the objective space (Miettinen 1998; He et al. 2021).

In this paper, we investigate the computation of the nadir objective vector for the multi-objective discrete optimization problem (MODOP). The MODOP can be written as

$$\begin{aligned} \min. \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^\intercal, \\ \text{s.t.} \quad & \mathbf{x} \in X, \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^\intercal$ is the decision vector (also called solution), and $X \subset \mathbb{Z}^n$ denotes the feasible set. $\mathbf{f} : \mathbb{Z}^n \to \mathbb{Z}^m$ is composed of m objective functions, and $\mathbf{f}(\mathbf{x})$ is the objective vector corresponding to $\mathbf{x}$.

Definition 1. *Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, $\mathbf{u}$ is said to* **dominate** *$\mathbf{v}$ (denoted as $\mathbf{u} \prec \mathbf{v}$), if and only if $u_i \leq v_i$ for every $i \in \{1, \ldots, m\}$ and $u_j < v_j$ for at least one $j \in \{1, \ldots, m\}$.*

Definition 2. *A decision vector $\mathbf{x}^* \in X$ and the corresponding objective vector $\mathbf{f}(\mathbf{x}^*)$ are* **Pareto-optimal**, *if there is no $\mathbf{x} \in X$ such that $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{f}(\mathbf{x}^*)$.*

Definition 3. *The set of all Pareto-optimal solutions is called the* **Pareto set** *(PS), and the set of all Pareto-optimal objective vectors is called the* **Pareto front** *(PF).*

Definition 4. *The* **ideal objective vector** *$\mathbf{z}^{ide}$ is composed of the lower bounds of the PF, i.e., $z_i^{ide} = \min_{\mathbf{x} \in PS} f_i(\mathbf{x})$ for $i = 1, \ldots, m$. The* **nadir objective vector** *$\mathbf{z}^{nad}$ consists of the upper bounds of the PF,* i.e., *$z_i^{nad} = \max_{\mathbf{x} \in PS} f_i(\mathbf{x})$ for $i = 1, \ldots, m$.*

Definition 5. *For the i-th objective function f_i with $i \in \{1, \ldots, m\}$, an objective vector $\mathbf{z}^{(i)^c}$ is called the* **critical point** *of f_i if it is Pareto-optimal and has the worst f_i value,* i.e., *$\mathbf{z}^{(i)^c} \in \{\mathbf{f}(\mathbf{x}) | \mathbf{x} = \arg\max_{\mathbf{x} \in PS} f_i(\mathbf{x})\}$.*

The nadir objective vector is often determined by obtaining the critical point on each objective, since $z_i^{nad} = z_i^{(i)^c}$ for each $i \in \{1, \ldots, m\}$. We propose a novel exact algorithm utilizing boundary decomposition (Zheng and

*Corresponding author

Symbol	Description
m (n)	The number of objectives (or variables).
$\mathbf{z}^{nad}$ ($\mathbf{z}^{ide}$)	The nadir (or ideal) objective vector.
$\mathbf{z}^{(i)^c}$	The critical point for the i-th objective.
$\mathbf{z}$ ($\tilde{\mathbf{z}}$)	A (transformed) objective vector; $\mathbf{z} = \mathbf{f}(\mathbf{x})$.
$\mathbf{w}$ ($\mathbf{w}^i$)	A boundary weight vector (of the i-th objective).
$\alpha, \mathbf{z}^r$	The parameter and the reference point in Eq. (4).
M	A finite trade-off in Definition 8.
μ	The user-defined trade-off between objectives.
$\mathbf{z}^{r1}, \mathbf{z}^{r2}$	Lower and upper bounds for normalization.
$\mathbf{y}$ ($\tilde{\mathbf{y}}$)	A (transformed) normalized objective vector; $\mathbf{y} = \mathbf{f}'(\mathbf{x})$ is obtained by Eq. (8).
$\boldsymbol{\phi}$	The projection of $\tilde{\mathbf{y}}$ onto $\mathbb{R}^{m-1}$ with some element removed; $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_{m-1})^\intercal$.
R	A rectangle; $R = \{\mathbf{p}, \mathbf{q}\}$.
L	The tuple of rectangles.

Table 1: Notation used in the paper.

Wang 2024) to find critical points for the MODOP with any number of objective functions. We term this algorithm as boundary decomposition for nadir objective vector computation (BDNC). The critical point of each objective can be found by optimizing a particular boundary subproblem under mild conditions (see the ***appendix*** for details). BDNC aims to solve m bilevel optimization problems to find these particular boundary subproblems. In our experimental studies, BDNC is tested on 4 MODOPs comprising a total of 720 instances. The results demonstrate that BDNC exhibits an overall lower and more stable runtime than baseline methods. Our contributions are listed as follows:

- Bilevel optimization problem based on boundary decomposition. We formulate a suitable bilevel optimization problem for each MODOP's objective, aiming to identify the particular boundary subproblem by comparing the optimal solutions of boundary subproblems.
- Exact algorithm for solving proposed bilevel optimization problems. The upper-level optimization performs a novel and effective pruning method for an exhaustive search in the $(m-1)$-dimensional space. The pruning method repeatedly subdivides the search space into rectangles and discards the invalid ones until all are classified as invalid. The lower-level optimization can adopt any suitable existing exact solver. BDNC guarantees convergence within a finite number of iterations.

A summary of symbols in this paper is listed in Table 1 for ease of reference. The appendix and the source code are available at https://github.com/EricZheng1024/BDNE.

Preliminaries

This section begins with introducing boundary decomposition (Zheng and Wang 2024). Subsequently, m boundary-decomposition-based bilevel optimization problems are constructed in a similar manner to align with the nadir objective vector.

Definition 6 (From (Ramirez-Atencia, Mostaghim, and Camacho 2017))**.** *Let $\tilde{\mathbf{z}}$ be the image of an objective vector $\mathbf{z}$ (denote $\tilde{\mathbf{z}}$ as the* **transformed objective vector***), where*

$$\tilde{z}_i = (1-\alpha)z_i + \frac{\alpha}{m}\sum_{j=1}^{m} z_j \ \text{ for } \ i = 1, \ldots, m. \quad (2)$$

Given a value of α ($0 < \alpha < 1$) and two objective vectors $\mathbf{u}$ and $\mathbf{v}$, $\mathbf{u}$ is said to **cone-dominate** $\mathbf{v}$ *(denoted as $\mathbf{u} \prec^c \mathbf{v}$) if and only if $\tilde{\mathbf{u}} \prec \tilde{\mathbf{v}}$.*

Definition 7. *Given a value of α, an objective vector is cone-optimal if no objective vector can cone-dominate it.*

Definition 8 (From (Geoffrion 1968))**.** *A decision vector $\mathbf{x}^*$ and the corresponding objective vector $\mathbf{f}(\mathbf{x}^*)$ are* **properly Pareto-optimal** *if they are Pareto-optimal and if there exists a finite number $M > 0$ such that, for each i and any $\mathbf{x} \in \Omega$, we have*

$$\frac{f_i(\mathbf{x}^*) - f_i(\mathbf{x})}{f_j(\mathbf{x}) - f_j(\mathbf{x}^*)} \leq M, \quad (3)$$

where j satisfies $f_j(\mathbf{x}^*) < f_j(\mathbf{x})$.

Boundary Subproblem

The boundary subproblem for the i-th objective is defined as

$$g_i^{bd}(\mathbf{x}|\mathbf{w}^i, \mathbf{z}^r, \alpha) = \max_{1\leq j\leq m} \left\{ w_j^i \left((1-\alpha) f_j(\mathbf{x}) + \frac{\alpha}{m}\sum_{k=1}^{m} f_k(\mathbf{x}) - z_j^r \right) \right\}, \quad (4)$$

where $\mathbf{w}^i$ is called the boundary weight vector of the i-th objective, $\mathbf{z}^r$ is a reference point, and $0 < \alpha < 1$. $\mathbf{w}^i$ satisfies three conditions: $w_i^i = 0; \forall j \in \{1, \ldots, m\}, w_j^i \geq 0; \exists\, j \in \{1, \ldots, m\}, w_j^i > 0$. The boundary subproblem involves two steps: 1) transforming the objective vector as shown in Eq. (2) and 2) scalarizing the transformed objective vector with the boundary weight vector. The boundary subproblem possesses some properties (see the ***appendix*** for details) that facilitate the search for the nadir objective vector. $\mathbf{z}^{(i)^c}$ can be determined by solving the boundary subproblem with a particular setting of $\mathbf{w}^i$ when $\alpha \leq \frac{m}{(m-1)M+m}$ where M is associated with the critical point. This is trivial as α can be a sufficiently small value. Besides, the optimal solution of the boundary subproblem must be Pareto-optimal.

Formulation of Bilevel Optimization Problem

We can search the critical points over the PF via the boundary subproblem. A previous proposed bilevel optimization problem (BLOP) is presented in the ***appendix***. The upper-level optimization problem (ULOP) aims to determine the particular boundary weight vector, while the lower-level optimization problem (LLOP) seeks the optimal solution of a potential boundary subproblem for the ULOP. To facilitate the design of the algorithm for the MODOP, we formulate an equivalent BLOP with $\mathbb{R}^{m-1}$ as the search space of its ULOP. According to Theorems 4 and 6 of the ***appendix***, the equivalent BLOP with respect to the i-th objective can be derived as

$$\begin{aligned} \max. \quad & f_i^b(\mathbf{u}, \mathbf{x}^*) = f_i(\mathbf{x}^*), \\ \text{s.t.} \quad & u_i = 0, \quad \forall j \neq i, z_j^r < u_j \leq u_j^{max}, \\ & \mathbf{x}^* = \arg\min_{\mathbf{x}\in X} g_i^{bd}(\mathbf{x}|\mathbf{w}^i, \mathbf{z}^r, \alpha), \end{aligned} \quad (5)$$

where $\mathbf{u}^{max}$ is the objective vector dominated by $\tilde{\mathbf{z}}^{nad}$ (*i.e.*, the transformed nadir objective vector), which can be set to

$$u_j^{max} = (1-\alpha)\max_{\mathbf{x}\in X} f_j(\mathbf{x}) + \frac{\alpha}{m}\sum_{k=1}^{m}\max_{\mathbf{x}\in X} f_k(\mathbf{x}), \quad (6)$$

and

$$w_i^i = 0, \quad \forall j \neq i, w_j^i = \frac{\beta}{u_j - z_j^r}. \quad (7)$$

It indicates that the search for the ULOP is conducted in the projection of transformed objective vectors onto $\mathbb{R}^{m-1}$ with the i-th component removed.

Moreover, let $\alpha = \frac{m}{(m-1)\mu+m}$, where $\mu > 0$ represents the upper bound of the trade-off between the k-th and l-th objectives, where $k = \arg\max_{1\leq j\leq m} f_j(\mathbf{x}^*) - f_j(\mathbf{x})$ and $l = \arg\max_{1\leq j\leq m} f_j(\mathbf{x}) - f_j(\mathbf{x}^*)$. In other words, μ is the amount of increment in the value of one objective function that the decision-maker is willing to tolerate in exchange for a one-unit decrement in another objective function. If the preference of decision-makers about μ is available, Corollary 3 of the ***appendix*** can guarantee to obtain a satisfactory nadir objective vector. When the preference is not provided, μ can take a sufficiently large value.

Exact Algorithm

We propose an exact algorithm called BDNC to solve the m BLOPs (*i.e.*, Problem (5)) sequentially or in parallel. The main procedure of BDNC is given in Algorithm 1. The main contribution of BDNC is its search method for the ULOP. Each LLOP is a scalarized problem, which can be solved by any suitable single-objective exact optimizer. All proofs in this section are presented in the ***appendix***.

Initialization

Firstly, an exact solver is selected and configured, which can match the properties of the concrete LLOP. $\mathbf{z}^{r1}$ and $\mathbf{z}^{r2}$ are two reference points for normalization, defined as $\mathbf{z}^{ide}$ and $\mathbf{z}^{max}$ respectively in this paper, where $z_j^{max} = \max_{\mathbf{x}\in X} f_j(\mathbf{x})$ for $j = 1,\ldots,m$. $\mathbf{z}^e$ is a temporary nadir objective vector, initialized to the smallest values. Thereafter, α of the BLOP is calculated according to μ. L represents the potential region of the search space. The potential region is initially set to the entire search space.

Normalization. M in the Definition 8 is affected by different value ranges of objective functions. The objective space should be normalized properly. Therefore, we introduce two reference points, namely $\mathbf{z}^{r1}$ and $\mathbf{z}^{r2}$, to normalize the objective space as

$$y_k = f'_k(\mathbf{x}) = \frac{f_k(\mathbf{x}) - z_k^{r1}}{z_k^{r2} - z_k^{r1}} \text{ for } k = 1,\ldots,m, \quad (8)$$

where $z_j^{r2} > z_j^{r1}$ for $j = 1,\ldots,m$ and $\mathbf{y}$ or $\mathbf{f}'(\mathbf{x})$ is a normalized objective vector. We let $z_j^r = 0$ for $j = 1,\ldots,m$ in Eq. (4). Then $\mathbf{z}^{r1}$ should be set according to Theorem 3 and Corollary 2. For example, it is reasonable to set $\mathbf{z}^{r1}$ to $\mathbf{z}^{ide}$ since $z_j^{ide} \leq f_j(\mathbf{x})$ for $j = 1,\ldots,m$ such that $(1-\alpha)f'_j(\mathbf{x}) + \frac{\alpha}{m}\sum_{k=1}^{m} f'_k(\mathbf{x}) \geq 0$.

Algorithm 1: BDNC

Input: An MODOP, μ; **Output**: $\mathbf{z}^e$.

```
Configure the exact solver of the LLOP.
Determine z^{r1} and z^{r2}; z^e ← z^{r1}.
α ← m/((m−1)μ+m); L ← ({0,1}).
for i = 1,...,m do
    while L ≠ ∅ do
        Prune the rectangles in L sequentially until a suitable rectangle R_s is selected.
        Solve the LLOP with the boundary weight vector associated with the maximal vertex of R_s.
        Determine the invalid region according to whether a new Pareto-optimal objective vector is generated.
        Remove rectangles in L within the invalid region.
    end while
end for
```

We consider that the search space of the ULOP is within an $(m-1)$-dimensional unit hypercube (*i.e.*, $(0,1]^{m-1}$). The settings of $\mathbf{z}^{r2}$ should ensure that all $\tilde{\mathbf{z}}^{(i)^c}$ are included in the rectangle defined by $\tilde{\mathbf{z}}^{r1}$ and $\tilde{\mathbf{z}}^{r2}$. We can easily deduce that: $\mathbf{z}' \in \{\mathbf{z}|z_j^{r1} < z_j \leq z_j^{r2}, j = 1,\ldots,m\}$ is a sufficient but unnecessary condition for $\tilde{\mathbf{z}}' \in \{\tilde{\mathbf{z}}|\tilde{z}_j^{r1} < \tilde{z}_j \leq \tilde{z}_j^{r2}, j = 1,\ldots,m\}$. Therefore, $\mathbf{z}^{r2}$ should satisfy $z_j^{r2} > z_j^{nad}$ for $j = 1,\ldots,m$. For example, $z_j^{r2} = \max_{\mathbf{x}\in X} f_j(\mathbf{x})$ for $j = 1,\ldots,m$. Furthermore, some additional suggestions can be found in (He, Ishibuchi, and Srinivasan 2021). For convenience, we introduce the rest of the algorithm, assuming a normalized objective space.

Invalid and Potential Regions. Solving the i-th BLOP yields the cone-optimal objective vector with the highest value of the i-th element according to Corollary 3. We consider that a region of the ULOP's search space is invalid if the transformed and projected result of this objective vector lies outside this region. Otherwise, it is considered potential. The main idea of optimizing the ULOP is to prune potential regions until no regions in the search space remain potential. In other words, eliminate the invalid regions until the removed regions collectively cover the entire search space. Let $\mathbf{w}^i$ be the boundary weight vector obtained by Eq. (7), $\mathbf{y}^*$ be the optimal objective vector to the boundary subproblem with $\mathbf{w}^i$, and $\boldsymbol{\phi}^*$ be the projection of $\tilde{\mathbf{y}}^*$. Propositions 1 and 2 are derived to determine the invalid and potential regions. The invalid regions in the two propositions are two $(m-1)$-dimensional axis-aligned rectangles. When there are numerous optimal objective vector to boundary subproblems, using the $(m-1)$-dimensional search space allows for a series of non-overlapping rectangles to represent potential or invalid regions rather than complex shapes, as shown in Figure 1. We use a tuple of rectangles, namely L, to represent the potential region. No optimal objective vector is available at the outset. As a result, L initially includes only the rectangle representing the entire search space.

Proposition 1. $\{\boldsymbol{\phi}|\phi_j^* \leq \phi_j \leq 1, j = 1,\ldots,m-1\}$ *is invalid.*

Proposition 2. *Let* $\mathbf{w}'$ *be the result of removing the* i*-th element from* $\mathbf{w}^i$ *and then* $l = \arg\max_{1\leq j\leq m-1}\{w'_j\phi_j^*\}$.

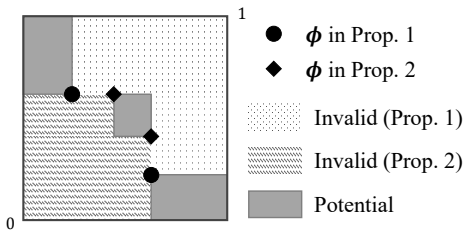

Figure 1: Illustration of potential and invalid regions.

$\{\boldsymbol{\phi}|0 < \phi_j < (\phi_l^* w_l')/w_j', j = 1, \ldots, m-1\}$ *is invalid.*

Search Procedure

Lines 5 to 10 in Algorithm 1 are the search steps for each objective. The steps are the same, except for the operating objective. A rectangle can be determined by two representative vertices. Then we can denote a rectangle as $R = \{\mathbf{p}, \mathbf{q}\}$ where $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{m-1}$ are minimal and maximal vertices of R respectively. The boundary weight vector $\mathbf{w}^i$ corresponding to R is defined as

$$w_l^i = \frac{1}{q_j} \quad \text{for} \quad j = 1, \ldots, m-1, \tag{9}$$

where $l = j$ if $j < i$ and $l = j + 1$ otherwise. The selection of $\mathbf{q}$ ensures that the nonzero elements in $\mathbf{w}^i$ are finite.

In line 6, each rectangle in L is associated with at least one pruning step. A pruning step refers to employing its corresponding invalid region to subdivide the rectangle(s) into many smaller rectangles and discarding those within the invalid region. On the one hand, the complexity of this procedure increases with the number of rectangles. On the other hand, a rectangle with many pruning steps can exhibit high complexity in its subdivision. Some rectangles in L may be invalid, but we cannot identify them before solving their corresponding boundary subproblems. Pruning these invalid rectangles and solving their associated boundary subproblems leads to a waste of computational resources. To reduce the overall complexity, we propose a lazy method to prune the search space in line 6, as detailed below:

Step 1 Select the last rectangle from the rectangles with most pruning steps in L and denote this rectangle as R_t=$\{\mathbf{p}^t, \mathbf{q}^t\}$ and the index of R_t in L as t.

Step 2 Let $l = j$ if $j < i$ and $l = j + 1$ otherwise. If there exists a decision vector $\mathbf{x} \in X$ such that

$$f_i(\mathbf{x}) > z_i^e \quad \text{and}$$
$$p_j^t \leq (1-\alpha) f_l'(\mathbf{x}) + \frac{\alpha}{m} \sum_{k=1}^{m} f_k'(\mathbf{x}) \leq q_j^t \tag{10}$$
$$\text{for } j = 1, \ldots, m-1,$$

L' stores the rectangles obtained by conducting all pruning steps in the step list of R_t sequentially. Otherwise, $L' \leftarrow \emptyset$.

Step 3 $L \leftarrow (L[1:t-1], L', L[t+1:end])$ and then $R_s \leftarrow L'[end]$.

Step 4 If $L' = \emptyset$ and $L \neq \emptyset$, go to Step 1. If $L' = \emptyset$ and $L = \emptyset$, the search procedure for the i-th objective is finished. Otherwise, return $R_s = \{\mathbf{p}^s, \mathbf{q}^s\}$.

In Step 1, we prioritize pruning the rectangles with the most steps to prevent them from accumulating too many steps. This prioritization rule can also be replaced by any other rule. The feasibility check in Step 2 aims to determine whether a solution belonging to the rectangle has a higher value for the i-th objective function. The negative result signifies the rectangle is invalid or one of the optimal solutions to the BLOP is found. In Step 3, L' is placed at the original position of R_t. This is because the rectangles in L are arranged in lexicographic order, prioritizing the first objective, then the second, and so forth. This lexicographic ordering can be completed automatically during the rectangle subdivision procedure without incurring additional computational costs. Besides, we improve the rectangle subdivision procedure described in (Kirlik and Sayın 2015) by subdividing only the rectangles that intersect with the invalid region.

Subsequently, a Pareto-optimal solution is obtained in line 7 and a new invalid region can be determined based on the obtained solution in line 8. If its corresponding objective vector is not obtained previously, a new invalid region can be identified based on Proposition 1. We can always determine a new invalid region according to Proposition 2 after solving the LLOP. However, it increases the complexity since two pruning steps may be added in an iteration. Therefore, we establish Proposition 3 upon the foundation of Proposition 2. The invalid region is given by Proposition 3 only in the absence of a new Pareto-optimal objective vector. In this way, only one invalid region is determined in each iteration. After that, the pruning step associated with this new invalid region is appended to the step lists of existing rectangles in L. In line 9, some invalid rectangles can be removed based on the recently determined invalid region.

Proposition 3. $\{\boldsymbol{\phi}|0 < \phi_j < q_j^s, j = 1, \ldots, m-1\}$ *is invalid if solving the boundary subproblem corresponding to the selected rectangle* $R_s = \{\mathbf{p}^s, \mathbf{q}^s\}$ *generates a previously obtained objective vector.*

An exhaustive search is achieved for the i-th objective when L is empty. BDNC terminates eventually as given by Theorem 1 and then outputs the nadir objective vector or the preferred one stated in Corollary 3.

Theorem 1. *BDNC converges in a finite number of iterations if the* PF *is finite.*

Example of Search Procedure

To better understand the behavior of BDNC, we illustrate the search procedure for the i-th objective using a straightforward example problem in Figure 2. The example problem is an MODOP with 3 objectives and 4 feasible solutions. The transformed and projected results of the feasible objective vectors are shown in the right part of the figure."1", "3", and "4" correspond to the Pareto-optimal solutions. "1" corresponds to the critical point of the i-th objective. "2" $>$ "1" $>$ "4" $>$ "3" in terms of the i-th objective function value. The first row of the figure highlights the selected rectangle

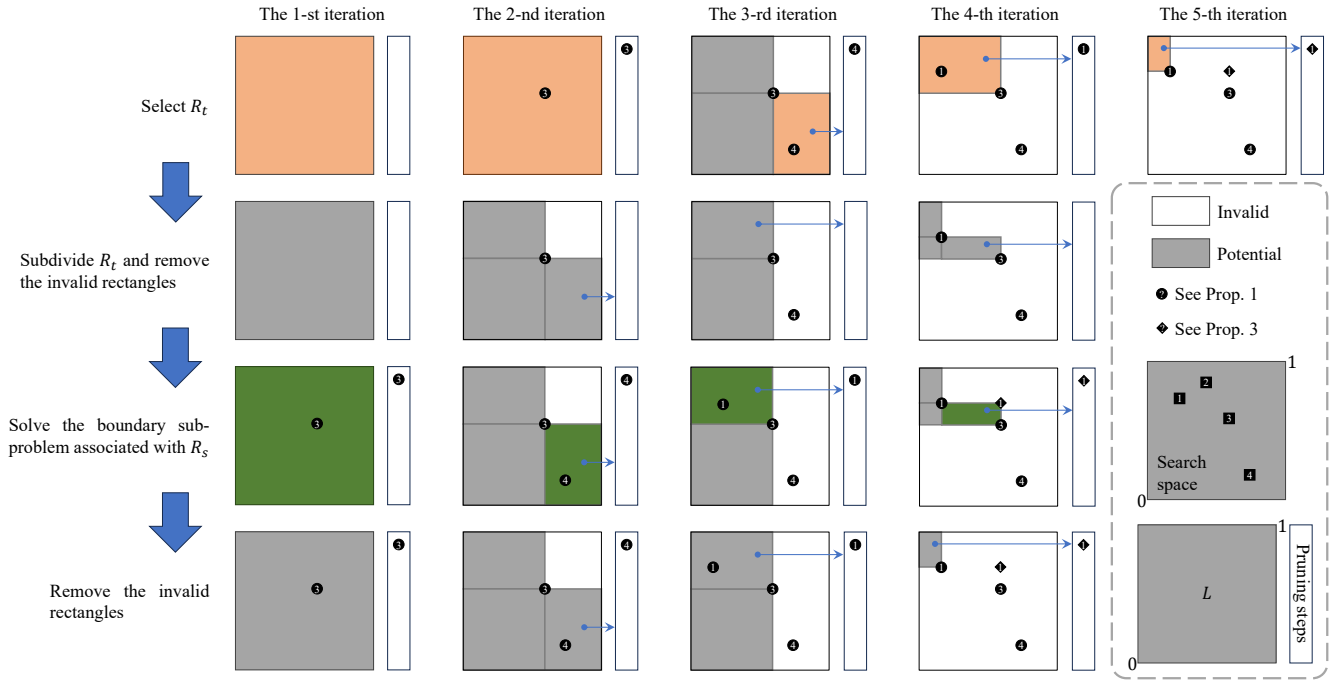

Figure 2: Example of the search procedure. The orange and green highlights denote the selected rectangles in their respective steps. The blue arrow indicates the rectangle to which the displayed pruning steps correspond.

in the first loop of line 6. The second row exhibits the final rectangles in line 6. The third row represents lines 7 and 8 and the last row shows line 9.

1**-st Iteration.** L is the whole search space initially. After the 1-st iteration, a new solution is obtained, and its corresponding pruning step is added to the step list of the sole rectangle R_s. The rectangle is not pruned immediately due to BDNC's lazy mechanism for handling the pruning steps.

2**-nd Iteration.** The only rectangle in L is subdivided at the 2-nd iteration, and four rectangles are present. The top right rectangle is removed since the invalid region defined by Dot "3" (*i.e.*, the upper right region of "3") covers this rectangle. A new solution is also obtained. Then, its corresponding pruning step is added to three step lists, as three rectangles are in L.

3**-rd Iteration.** All the rectangles have the same number of pruning steps, and the three rectangles are arranged in lexicographic order. Then, the bottom right one is examined first at the beginning of the 3-rd iteration. However, this rectangle does not contain the solution satisfying Constraints (10). This rectangle is discarded and the top left one containing "1" and "2" is pruned. The selected rectangle remains unchanged after pruning because the invalid region defined by Dot "4" (*i.e.*, the upper right region of "4") does not intersect with it. At the end of the 3-rd iteration, the critical point is achieved. But L is not empty and BDNC should examine the remaining rectangles.

4**-th Iteration.** The bottom rectangle has 2 pruning steps, including Dots "4" and "1". The top rectangle has 1 pruning step but contains "2". As a result, the bottom rectangle is discarded directly and the top one is pruned. Since no new solution is obtained, the invalid region is determined according to Proposition 3. Two rectangles within the invalid region (*i.e.*, the lower left region of Rhombus "1") are removed.

5**-th Iteration.** Since the only remaining rectangle does not contain a solution with a higher value of the i-th objective function, the rectangle is removed, leaving L empty at this point. Therefore, the search for this objective is complete.

Related Works

To contextualize our work, we review the most relevant literature on identifying the nadir objective vector.

Heuristic Method. Heuristic methods can provide an (initial) estimate of the nadir objective vector. Although heuristic methods are simple, they lack theoretical guarantees and may produce significant errors for certain problems. More details of heuristic methods can be found in the ***appendix***.

Exact Method. The exact method formulates a series of optimization problems based on the original MOP. Solving the defined optimization problems guarantees to find the nadir objective vector. When the exact solver is available, these methods converge within a finite number of iterations. Ehrgott and Tenfelde-Podehl (2003) propose to obtain the Pareto-optimal solutions of the original m-dimensional MOP from the set of Pareto-optimal solutions of the m $(m-1)$-dimensional MOPs. However, this method is exceedingly computationally inefficient. After that, Kirlik and Sayın (2015) formulate m BLOPs, where each ULOP involves an $(m-2)$-dimensional search space and the LLOP uses the ϵ-constraint scalarization method. Compared with

Problem	m	n	KS			KL			FD&IS			BDNC	
			median	Δ	n_{TO}	median	Δ	n_{TO}	median	Δ	n_{TO}	median	n_{TO}
MOAP	3	20×20	118.9(3)-	95.6	0	107(2)-	83.8	0	123.5(4)-	100.3	0	23.2(1)	0
	4	10×10	250.3(3)-	227.4	0	21.2(1)=	-1.7	0	4445(4)-	4422.1	1	22.9(2)	0
	5	10×10	TO(4)-	TO	30	214.1(1)=	-51.7	0	TO(4)-	TO	30	265.7(2)	0
	3	30×30	357.9(3)-	165.6	0	1518.4(4)-	1326.2	3	342.3(2)-	150	0	192.3(1)	0
	4	15×15	4637.3(3)-	4473.9	0	627.8(2)-	464.4	0	TO(4)-	TO	30	163.4(1)	0
	5	15×15	TO(4)-	TO	30	TO(4)-	TO	30	TO(4)-	TO	30	5095.5(1)	3
MOKP	3	200	1759.8(4)-	1663.2	0	142.4(2)-	45.8	0	294.2(3)-	197.6	0	96.6(1)	0
	4	50	1525(3)-	1487.9	0	39.7(2)=	2.6	0	TO(4)-	TO	19	37.1(1)	0
	5	50	TO(4)-	TO	30	TO(4)-	TO	24	TO(4)-	TO	30	542.3(1)	0
	3	240	3999.7(4)-	3842	0	244.7(2)-	87	0	391.9(3)-	234.2	0	157.7(1)	0
	4	60	3651.3(3)-	3590.2	5	89.8(2)-	28.7	0	TO(4)-	TO	29	61.1(1)	0
	5	60	TO(4)-	TO	30	TO(4)-	TO	30	TO(4)-	TO	30	1264.2(1)	0
MOILP	3	80	37.3(3)-	18.7	0	13.9(1)=	-4.6	0	42.4(4)-	23.9	0	18.6(2)	0
	4	40	291.3(3)-	271.6	1	22(2)=	2.3	0	2970.1(4)-	2950.4	8	19.7(1)	0
	5	40	TO(4)-	TO	28	928.7(2)-	670.4	5	TO(4)-	TO	30	258.4(1)	0
	3	100	120.8(4)-	74.8	0	34.3(1)+	-11.8	0	97.6(3)-	51.5	0	46.1(2)	0
	4	50	1063.1(3)-	1030.1	4	64.1(2)-	31	0	7126.7(4)-	7093.6	11	33.1(1)	0
	5	50	TO(4)-	TO	30	TO(4)-	TO	16	TO(4)-	TO	30	469.9(1)	0
MOSPP	3	50×50	37.9(2)-	11.1	0	52.4(3)-	25.5	0	74.8(4)-	48	0	26.8(1)	0
	4	25×25	71.8(3)-	50.9	0	16.6(1)+	-4.3	0	742.5(4)-	721.6	0	20.9(2)	0
	5	25×25	2253.9(3)-	2118.6	0	586.6(2)-	451.3	5	TO(4)-	TO	30	135.3(1)	0
	3	60×60	115.1(2)-	57.4	0	157.3(3)-	99.5	0	172.7(4)-	115	0	57.7(1)	0
	4	30×30	216(3)-	163.1	0	58.8(2)=	6	0	1961.4(4)-	1908.5	0	52.8(1)	0
	5	30×30	TO(4)-	TO	17	TO(4)-	TO	28	TO(4)-	TO	30	412.4(1)	0
Total +/=/-			0/0/24			2/6/16			0/0/24			NA	
Average rank			2.8			2.2			3.8			1.2	

Table 2: Median runtimes in seconds. "Δ" denotes the gap of median runtimes between the baseline and NDNC. "n_{TO}" denotes the number of instances where the algorithm exceeds the time limit. "TO" represents timeout. The rank of each problem is shown after the median runtime. "+", "=" or "-" denotes that the performance of the corresponding algorithm is statistically better than, similar to, or worse than that of BDNC based on Wilcoxon's rank sum test at 0.05 significant level.

BDNC, this method has a more time-consuming rectangle subdivision procedure, and its LLOP involves two sequential ϵ-constraint-based optimization problems. Köksalan and Lokman (2015) propose two sequential optimization problems. The method eliminates the need for bilevel optimization and instead involves solving these problems multiple times. Each time after solving the first problem, the optimal solution of the second problem determines the additional variables and constraints for the first one. The efficiency of this method is significantly affected by an increasing number of variables and constraints. Özpeynirci (2017) introduces a two-stage framework to reduce the iteration number of Köksalan and Lokman's method. Firstly, many facets, defining convex hulls, are detected. Then, Köksalan and Lokman's method is applied to search the critical points inside each convex hull. Nevertheless, the improvement of the framework is not well studied in the experiments of (Özpeynirci 2017). Moreover, some exact methods are specifically designed for a subset of the MODOP, such as the multi-objective integer linear program (Boland, Charkhgard, and Savelsbergh 2017). An additional advantage of BDNC over existing exact methods is its capability to accommodate flexible trade-off settings specified by the decision-maker. A comparison of our methods to existing methods is summarized in the ***appendix***.

Experiments

Experimental Setup

The multi-objective assignment problem (MOAP), the multi-objective knapsack problem (MOKP), the general multi-objective integer linear programming (MOILP), and the multi-objective shortest path problem (MOSPP) are utilized in the experimental studies. 24 test problems with different numbers of objectives and variables are considered as shown in Table 2. 30 instances are randomly generated for each problem. The random instance generation schemes for the MOKP, the MOAP, and the MOILP are the same as those in (Kirlik and Sayın 2015). The generation scheme for the MOSPP follows (Lokman 2007).

BDNC with $\mu = 10^6$ is compared with 3 state-of-the-art algorithms: Kirlik and Sayın (2015)'s method denoted as KS, Köksalan and Lokman (2015)'s method denoted as KL, and FD&IS (Özpeynirci 2017). Algorithms are implemented on MATLAB R2022a. The scalarized problem is solved by the built-in function "intlinprog". Experiments are executed on a computer equipped with two 3.00-GHz Intel Xeon Gold

Problem	m	n	KS		KL		FD&IS		BDNC	
			best	worst	best	worst	best	worst	best	worst
MOAP	3	20×20	75.6(4)	177(2)	45.5(2)	214.5(4)	62.2(3)	194.4(3)	14.3(1)	29.4(1)
	4	10×10	103.7(3)	445.9(3)	11.2(1)	51.9(2)	1680.1(4)	TO(4)	12(2)	39.3(1)
	5	10×10	TO(4)	TO(4)	79.7(1)	1730.2(2)	TO(4)	TO(4)	124.5(2)	525.7(1)
	3	30×30	284.3(4)	432.3(2)	162.8(2)	TO(4)	166(3)	727.1(3)	125.9(1)	263.1(1)
	4	15×15	2231.5(3)	8027.9(3)	320.4(2)	1603.8(2)	TO(4)	TO(4)	90.2(1)	298.4(1)
	5	15×15	TO(4)	TO(4)	TO(4)	TO(4)	TO(4)	TO(4)	1523.8(1)	TO(4)
MOKP	3	200	1058.4(4)	3078.5(4)	76.6(2)	436.9(2)	163.4(3)	597.2(3)	50.8(1)	199.2(1)
	4	50	411.5(3)	4094(3)	20.7(2)	162.2(2)	4043.3(4)	TO(4)	15.3(1)	80.2(1)
	5	50	TO(4)	TO(4)	1527.7(2)	TO(4)	TO(4)	TO(4)	112.8(1)	3892.3(1)
	3	240	2840.5(4)	5680.3(4)	96.8(2)	794.9(2)	237.5(3)	796(3)	86.4(1)	286.6(1)
	4	60	833.8(3)	TO(4)	24.4(2)	256.1(2)	9899.5(4)	TO(4)	17.9(1)	117.9(1)
	5	60	TO(4)	TO(4)	TO(4)	TO(4)	TO(4)	TO(4)	465.7(1)	3533.2(1)
MOILP	3	80	11.8(3)	102.2(4)	4.9(1)	48.5(1)	20.4(4)	93.4(3)	6.7(2)	61.9(2)
	4	40	29.5(3)	TO(4)	3.7(1)	679.8(2)	856(4)	TO(4)	5.4(2)	77.6(1)
	5	40	152.8(3)	TO(4)	4(1)	TO(4)	TO(4)	TO(4)	14.1(2)	977.7(1)
	3	100	14.6(3)	372.3(4)	8.4(1)	248(1)	35.8(4)	346.8(3)	9.2(2)	262.3(2)
	4	50	56.1(3)	TO(4)	7.1(2)	824.1(2)	521.8(4)	TO(4)	7(1)	149.7(1)
	5	50	TO(4)	TO(4)	42.3(1)	TO(4)	TO(4)	TO(4)	82.7(2)	3221.6(1)
MOSPP	3	50×50	15.8(3)	92.6(2)	15.4(2)	119.3(3)	40.9(4)	171.4(4)	14.1(1)	50.5(1)
	4	25×25	7.7(1)	183.4(3)	8.7(2)	45.7(2)	263(4)	2870.3(4)	9.9(3)	35.8(1)
	5	25×25	739.9(3)	5649(2)	62.7(1)	TO(4)	TO(4)	TO(4)	68.4(2)	265.6(1)
	3	60×60	49.4(2)	205.9(2)	73.3(3)	273.4(3)	87.5(4)	299.7(4)	34.6(1)	96.8(1)
	4	30×30	64.8(3)	626.9(3)	22.2(2)	115.5(2)	592(4)	8853.9(4)	17.6(1)	84.8(1)
	5	30×30	2536.6(2)	TO(4)	5400.9(3)	TO(4)	TO(4)	TO(4)	211.5(1)	907.9(1)

Table 3: Best and worst runtimes in seconds. "TO" represents timeout. The rank on each problem is shown after the runtime.

6248R CPUs and 128GB of RAM. The maximum runtime for an algorithm on each objective is 3600 seconds. The runtime in seconds is recorded as the performance metric.

Experimental Results

Table 2 reports the statistical results of BNDC on each problem. BDNC has the best average rank. BDNC significantly outperforms KS and FD&IS on all problems and surpasses KL across most cases. KL is the second-best algorithm. KL only has better performance than BDNC on the 3-objective MOILP and the 4-objective small-scale MOSPP, and the performance gaps are relatively small. The comparison algorithms are highly inefficient on the 5-objective problems. They exceed the time limit on most instances of the 5-objective large-scale problem. In contrast, BDNC encounters timeouts on only 3 instances of the large-scale MOAP with 5 objectives. Furthermore, we utilize Table 3 to demonstrate the stability of BDNC. The best runtimes of BDNC rank the first on 15 out of 24 problems, while those of KL rank the first on 8 problems. KL has a competitive performance in terms of the best runtime. Nevertheless, when examining the worst runtimes, BDNC demonstrates superior performance across most problems, indicating strong robustness. In addition, the performance of KS deteriorates remarkably as the number of objectives increases, due to the increasing number of Pareto-optimal solutions and the high complexity of its rectangle subdivision procedure. FD&IS achieves the worst performance as a variant of KL. It only shows a more stable performance than KL on the 3-objective MOAP. In general, Tables 2 and 3 validate the superior performance and stability of BDNC, respectively.

More experimental analyses are provided in the ***appendix***, including ablation studies, evaluations on other test problems, and investigations of the parameter μ.

Conclusion

This paper has proposed BDNC for the MODOP, having theoretical guarantees on identifying the nadir objective vector, ensuring finite-time convergence, and supporting the trade-off of decision-makers. BDNC adopts bilevel optimization based on boundary decomposition. The LLOP aims to determine the optimal solution to the boundary subproblem, solved by any appropriate single-objective exact solver. The ULOP seeks the boundary weight vector aligned with the critical point, addressed by our proposed method. The search space of the ULOP is represented by an $(m-1)$-dimensional rectangle. The search procedure utilizes a lazy pruning method, which subdivides the search space into many small rectangles and eliminates the invalid ones in each iteration. Experimental studies on various test instances have demonstrated that BDNC achieves superior overall performance and low runtime variance.

In future work, we will further investigate the pruning method to refine the overall and anytime performance of BDNC. Additionally, we plan to develop a robust version with parallel implementation and then connect it with decision-making systems.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62106096 and Grant No. 62476118), the Natural Science Foundation of Guangdong Province (Grant No. 2024A1515011759), the National Natural Science Foundation of Shenzhen (Grant No. JCYJ20220530113013031).

References

Boland, N.; Charkhgard, H.; and Savelsbergh, M. 2017. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *European Journal of Operational Research*, 260(3): 904–919.

Branke, J.; Branke, J.; Deb, K.; Miettinen, K.; and Slowiński, R. 2008. *Multiobjective optimization: Interactive and evolutionary approaches*, volume 5252. Springer.

Ehrgott, M.; and Tenfelde-Podehl, D. 2003. Computation of ideal and nadir values and implications for their use in MCDM methods. *European Journal of Operational Research*, 151(1): 119–139.

Geoffrion, A. M. 1968. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22(3): 618–630.

He, L.; Ishibuchi, H.; and Srinivasan, D. 2021. Metric for evaluating normalization methods in multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 403–411.

He, L.; Ishibuchi, H.; Trivedi, A.; Wang, H.; Nan, Y.; and Srinivasan, D. 2021. A survey of normalization methods in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 25(6): 1028–1048.

Kirlik, G.; and Sayın, S. 2015. Computing the nadir point for multiobjective discrete optimization problems. *Journal of Global Optimization*, 62: 79–99.

Köksalan, M.; and Lokman, B. 2015. Finding nadir points in multi-objective integer programs. *Journal of Global Optimization*, 62: 55–77.

Lokman, B. 2007. *Approaches for multi-objective combinatorial optimization problems*. Master's thesis, Middle East Technical University.

Mena, R.; Godoy, M.; Catalán, C.; Viveros, P.; and Zio, E. 2023. Multi-objective two-stage stochastic unit commitment model for wind-integrated power systems: A compromise programming approach. *International Journal of Electrical Power & Energy Systems*, 152: 109214.

Mesquita-Cunha, M.; Figueira, J. R.; and Barbosa-Póvoa, A. P. 2023. New ϵ-constraint methods for multi-objective integer linear programming: A Pareto front representation approach. *European Journal of Operational Research*, 306(1): 286–307.

Miettinen, K. 1998. *Nonlinear multiobjective optimization*. Springer.

Momma, M.; Dong, C.; and Liu, J. 2022. A multi-objective/multi-task learning framework induced by Pareto stationarity. In *International Conference on Machine Learning (ICML)*, 15895–15907. PMLR.

Özpeynirci, Ö. 2017. On nadir points of multiobjective integer programming problems. *Journal of Global Optimization*, 69: 699–712.

Ramirez-Atencia, C.; Mostaghim, S.; and Camacho, D. 2017. A knee point based evolutionary multi-objective optimization for mission planning problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 1216–1223.

Vodopija, A.; Tušar, T.; and Filipič, B. 2024. Characterization of constrained continuous multiobjective optimization problems: A performance space perspective. *IEEE Transactions on Evolutionary Computation*.

Wang, H.; Olhofer, M.; and Jin, Y. 2017. A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. *Complex & Intelligent Systems*, 3: 233–245.

Weerasuriya, A. U.; Zhang, X.; Wang, J.; Lu, B.; Tse, K. T.; and Liu, C.-H. 2021. Performance evaluation of population-based metaheuristic algorithms and decision-making for multi-objective optimization of building design. *Building and Environment*, 198: 107855.

Zhang, X.; Lin, X.; Xue, B.; Chen, Y.; and Zhang, Q. 2023. Hypervolume maximization: A geometric view of Pareto set learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 36.

Zheng, R.; and Wang, Z. 2024. Boundary Decomposition for Nadir Objective Vector Estimation. *Advances in Neural Information Processing Systems (NeurIPS)*.