A dual method for polar cuts in disjoint bilinear programming

Xiaosong Ding[1], **Jun Ma**[1], **Xi Chen**[1] and **Chao Liu**[1,*]

[1] *International Business School, Beijing Foreign Studies University, Beijing, P.R.China, 100089*
E-mail: ⟨{*dingxiaosong, 202220316002, chenxi0109, 202220316001*}*@bfsu.edu.cn*⟩

Abstract. As one branch of deterministic approaches to disjoint bilinear programming, cutting plane methods are renowned for its ability to systematically reduce the search space by adding cutting planes that are able to cut off regions deemed infeasible or suboptimal. Polar cuts have been widely utilized as a dominating type of cut in terms of deepness. During the establishment of a polar cut, the modified Newton's method is employed to derive the cutting points along the positive or negative extensions of edges emanating from a local solution. Nonetheless, its performance can be further improved along the positive extensions. Drawing inspiration from integer programming, we develop a new approach based on the LP duality theory for this purpose. It re-formulates the original program with a piece-wise linear concave objective function as a single LP. Moreover, we propose a new technique to derive the edges as accommodation to degeneracy. Numerical results show that, by utilizing our newly developed dual method, computing time can be gradually saved as the percentage of generated cutting points along the positive extensions of edges rises.

Keywords: degeneracy, disjoint bilinear programming, duality, LP, polar cuts

Received: January 30, 2024; accepted: March 3, 2024; available online: May 27, 2024

DOI: 10.17535/crorr.2024.0006

1. Introduction

With $\boldsymbol{X}_0$ and $\boldsymbol{Y}_0$ being two bounded and non-empty polytopes, traditional disjoint bilinear programming (DBLP) can be stated as

$$\begin{aligned} \min \quad & f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{c}^t\boldsymbol{x} + \boldsymbol{d}^t\boldsymbol{y} + \boldsymbol{x}^t\boldsymbol{C}\boldsymbol{y}, \\ \text{s.t.} \quad & \boldsymbol{x} \in \boldsymbol{X}_0 = \{\boldsymbol{x} \in \mathbb{R}^{n_1} : \boldsymbol{A}_1\boldsymbol{x} \le \boldsymbol{b}_1,\ \boldsymbol{x} \ge \boldsymbol{0}\}, \quad \boldsymbol{A}_1 \in \mathbb{R}^{m_1 \times n_1},\ \boldsymbol{b}_1 \in \mathbb{R}^{m_1}, \\ & \boldsymbol{y} \in \boldsymbol{Y}_0 = \{\boldsymbol{y} \in \mathbb{R}^{n_2} : \boldsymbol{A}_2\boldsymbol{y} \le \boldsymbol{b}_2,\ \boldsymbol{y} \ge \boldsymbol{0}\}, \quad \boldsymbol{A}_2 \in \mathbb{R}^{m_2 \times n_2},\ \boldsymbol{b}_2 \in \mathbb{R}^{m_2}. \end{aligned} \tag{1}$$

A large amount of literature exists on the solution approaches to DBLP (1), among which two major deterministic approaches are cutting plane methods and branch and bound methods.

Traditional cutting plane methods for DBLP (1) are based on its structural property that ensures the existence of global solution at a pair of vertices in $\boldsymbol{X}_0 \times \boldsymbol{Y}_0$. Concavity cuts were first developed in [15] and widely utilized in early developed cutting plane methods [3, 6, 10, 17]. Their convergence had been investigated in several papers [7, 9, 21]. However, it was shown that concavity cuts, as well as other types of cuts, are uniformly dominated by polar cuts employed in other cutting plane methods [13, 18]. Decomposition cuts were developed in order to extend the concept of concavity cuts for deeper cuts [8, 11, 12]. Some other relevant research can also be found; see for example [2, 4, 5, 14].

In a cutting plane method for DBLP (1), two parts are time-consuming. One part lies in its local optimization phase with the purpose of identifying a good local solution, $\overline{\boldsymbol{x}}$. In this part,

[*]Corresponding author.

if we borrow the idea from a local star minimum (also refer to **Definition 2**) to pursue a better candidate by examining $\overline{\boldsymbol{x}}$'s neighborhood, considerable computing time will elapse. Given the existence of a better candidate, meaning that the algorithm can escape successfully from the current local solution, the entire searching process will restart right from the beginning. Things can get even worse with more escapes. Unfortunately, nowadays, this is what all extant cutting plane methods exactly do and is still considered as the most efficient approach in search of $\overline{\boldsymbol{x}}$ for DBLP (1). The other part lies in the development of a cut, say, a polar cut. Given $\overline{\boldsymbol{x}}$ from the local optimization phase, for each edge emanating from $\overline{\boldsymbol{x}}$, we need to calculate a cutting point along either its positive or negative extension (will be referred to as positive or negative cutting point(s), respectively). In [13], the modified Newton's method was developed for this purpose. Drawing inspiration from integer programming, we develop herein a new method based on the LP duality theory for the derivation of positive cutting points. By re-formulating the original program with a piece-wise linear concave objective function as a single LP, it is possible to accelerate the generation of a polar cut and save computing time as the percentage of positive cutting points rises. Moreover, for the derivation of edges emanating from $\overline{\boldsymbol{x}}$, we propose a new approach being able to accommodate degeneracy.

The rest of this paper proceeds as below. Section 2 briefly describes the local optimization phase that can provide a candidate, $\overline{\boldsymbol{x}}$. Section 3 is devoted to the development of an augmented method for establishing a polar cut. Section 4 provides numerical results to evaluate the performance of our newly developed method, and the final section concludes our paper.

2. Local Optimization

The essential solution property of DBLP exploited in the local optimization phase of almost all cutting plane methods is that even though $f(\boldsymbol{x}, \boldsymbol{y})$ is not quasi-concave, the global optimizer, $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, is attained at a vertex of $\boldsymbol{X}_0 \times \boldsymbol{Y}_0$, which means that $\boldsymbol{x}$ and $\boldsymbol{y}$ are vertices of $\boldsymbol{X}_0$ and $\boldsymbol{Y}_0$, respectively [6].

To facilitate our presentation, denote by $\boldsymbol{X}_0^i$ the original feasible region $\boldsymbol{X}_0$ when $i = 0$, or its subset obtained after i cuts have been introduced.

Definition 1. *A local minimizer of $g(\cdot)$ over $\boldsymbol{X}_0^i$ is a vertex, $\boldsymbol{x}_{\ell m}$, such that $g(\boldsymbol{x}_{\ell m}) \leq g(\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathcal{B}_\delta(\boldsymbol{x}_{\ell m}) \cap \boldsymbol{X}_0^i$, where $\mathcal{B}_\delta(\boldsymbol{x}_{\ell m})$ is a δ-neighborhood around $\boldsymbol{x}_{\ell m}$ in $\boldsymbol{X}_0^i$, and $g(\boldsymbol{x}_{\ell m})$ is the corresponding local minimum.*

Definition 2. *A local star minimizer of $g(\cdot)$ over $\boldsymbol{X}_0^i$ is a vertex, $\boldsymbol{x}_{\ell sm}$, such that $g(\boldsymbol{x}_{\ell sm}) \leq g(\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathcal{N}_{\boldsymbol{X}_0^i}(\boldsymbol{x}_{\ell sm})$, where $\mathcal{N}_{\boldsymbol{X}_0^i}(\boldsymbol{x}_{\ell sm})$ denotes the vertices adjacent to $\boldsymbol{x}_{\ell sm}$ in $\boldsymbol{X}_0^i$, and $g(\boldsymbol{x}_{\ell sm})$ is the corresponding local star minimum.*

Since $f(\boldsymbol{x}, \boldsymbol{y})$ is not quasi-concave, a local star minimum is not necessarily a local minimum, and thus the development of a cut from a local star minimizer cannot take effect as usual for those with quasi-concave objective functions. Moreover, for DBLP (1), cuts involving variables associated with both $\boldsymbol{X}_0^i$ and $\boldsymbol{Y}_0$ may destroy their special structure, and thereby fail the existing efficient algorithms to solve sub-problems. As a result, to develop a cut that involves only the $\boldsymbol{x}$-variables and yet is convergent from a local minimizer, a concept more than **Definition 1, 2** is necessary [18].

Definition 3. *A vertex $(\overline{\boldsymbol{x}}^i, \overline{\boldsymbol{y}})$ in DBLP is a Pseudo-Global Minimizer (PGM) if $f(\overline{\boldsymbol{x}}^i, \overline{\boldsymbol{y}}) \leq f(\boldsymbol{x}, \boldsymbol{y})$ for each $\boldsymbol{x} \in \mathcal{B}_\delta(\overline{\boldsymbol{x}}^i) \cap \boldsymbol{X}_0^i$ and for each $\boldsymbol{y} \in \boldsymbol{Y}_0$.*

For DBLP (1), a vertex is adjacent to $(\overline{\boldsymbol{x}}^i, \overline{\boldsymbol{y}})$ if and only if it is either of the form $(\boldsymbol{x}^k, \overline{\boldsymbol{y}})$ or $(\overline{\boldsymbol{x}}^i, \boldsymbol{y}^k)$ where $\boldsymbol{x}^k \in \mathcal{N}_{\boldsymbol{X}_0^i}(\overline{\boldsymbol{x}}^i)$ and $\boldsymbol{y}^k \in \mathcal{N}_{\boldsymbol{Y}_0}(\overline{\boldsymbol{y}})$. For a PGM, further improvement may be achieved by an idea analogous to that suggested by **Definition 2**, i.e., we can examine those vertices adjacent to $\overline{\boldsymbol{x}}^i$ for a better solution. A so derived PGM can have the advantages from both a local minimum and a local star minimum. **Algorithm 1**, originated from [6] to identify a PGM, $(\overline{\boldsymbol{x}}^i, \overline{\boldsymbol{y}})$, is currently acting as a building block in the local optimization phase of a cutting plane method. It can be perceived that a PGM provided by **Algorithm 1** serves as a local minimizer in $\boldsymbol{X}_0^i$ and a global minimizer in $\boldsymbol{Y}_0$.

Algorithm 1: Augmented Mountain Climbing Method

```
  Input: $\boldsymbol{C}$, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{X}_{\boldsymbol{0}}^i$, $\boldsymbol{Y}_0$, $\widetilde{\boldsymbol{y}} \in \boldsymbol{Y}_0$.
  Output: $(\overline{\boldsymbol{x}}^i, \overline{\boldsymbol{y}})$.
1 repeat
2     $\widetilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \boldsymbol{X}_0^i} f(\boldsymbol{x}, \widetilde{\boldsymbol{y}})$;    $\widetilde{\boldsymbol{y}} = \arg\min_{\boldsymbol{y} \in \boldsymbol{Y}_0} f(\widetilde{\boldsymbol{x}}, \boldsymbol{y})$;
3 until $\widetilde{\boldsymbol{x}}$ converges;
4 construct $\mathcal{N}_{\boldsymbol{X}_0^i}(\widetilde{\boldsymbol{x}})$;
5 if $\exists \check{\boldsymbol{x}} \in \mathcal{N}_{\boldsymbol{X}_0^i}(\widetilde{\boldsymbol{x}})$ such that $f(\check{\boldsymbol{x}}, \boldsymbol{y}^*) = \min_{\boldsymbol{y} \in \boldsymbol{Y}_0} f(\check{\boldsymbol{x}}, \boldsymbol{y}) < f(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$ then
6     go to line 2 with $\widetilde{\boldsymbol{y}} = \boldsymbol{y}^*$;
7 end
8 terminate with $(\overline{\boldsymbol{x}}^i, \overline{\boldsymbol{y}}) = (\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$ as a PGM.
```

3. Polar Cuts

By **Algorithm 1**, we can reach a PGM, $(\overline{\boldsymbol{x}}^i, \overline{\boldsymbol{y}})$, and try to generate a valid cut from $\overline{\boldsymbol{x}}^i$ to cut off $\overline{\boldsymbol{x}}^i$ together with a portion of $\boldsymbol{X}_0^i$. This is referred to as the global optimization phase of a cutting plane method, following which **Algorithm 1** will restart within the reduced feasible region, $\boldsymbol{X}_0^{i+1}$. Basically, a good cut should eliminate as large as possible the current feasible region without excluding any potential global minimizer. To this end, polar cuts can serve as a well qualified candidate but may confront several computational issues during the generation. For a clear presentation, in what follows, we will simply take $\overline{\boldsymbol{x}}$ as $\overline{\boldsymbol{x}}^i$ in a PGM, $(\overline{\boldsymbol{x}}^i, \overline{\boldsymbol{y}})$.

3.1. Basics

Definition 4. *For DBLP (1), given a set* $\boldsymbol{Y}_0$ *and a scalar* α*, the generalized reverse polar of* $\boldsymbol{Y}_0$ *relative to* α *is the set*

$$\boldsymbol{Y}_0(\alpha) = \left\{ \boldsymbol{x} \in \mathbb{R}^{n_1} : \min_{\boldsymbol{y} \in \boldsymbol{Y}_0} f(\boldsymbol{x}, \boldsymbol{y}) \geq \alpha \right\}.$$

In $\boldsymbol{Y}_0(\alpha)$, the scalar α actually serves as the current best objective value of $f(\boldsymbol{x}, \boldsymbol{y})$. By [13] and [16], $\boldsymbol{Y}_0(\alpha)$ is a polytope. Denote by $\boldsymbol{y}^i$ and $\mathcal{H}_i^+$, $i = 1, 2, \ldots, u$, the u vertices of $\boldsymbol{Y}_0$ and the corresponding u closed half-spaces, respectively. We can rewrite $\boldsymbol{Y}_0(\alpha)$ as

$$\boldsymbol{Y}_0(\alpha) = \cap_{i=1}^u \mathcal{H}_i^+(\boldsymbol{x}) = \cap_{i=1}^u \left\{ \boldsymbol{x} \in \mathbb{R}^{n_1} : \boldsymbol{c}^t \boldsymbol{x} + \boldsymbol{d}^t \boldsymbol{y}^i + \boldsymbol{x}^t \boldsymbol{C} \boldsymbol{y}^i \geq \alpha \right\}.$$

Let x_j, $j \in \mathcal{J}$, be the n_1 non-basic variables at $\overline{\boldsymbol{x}}$, where $\mathcal{J}$ is the index set for non-basic variables. Given that $\overline{\boldsymbol{x}}$ is non-degenerate, there exist precisely n_1 distinct edges incident to $\overline{\boldsymbol{x}}$. Consider the extended simplex tableau in Tucker form corresponding to $\overline{\boldsymbol{x}}$, and let $\boldsymbol{e}^j$ be the extended column of the non-basic variable x_j with components representing the corresponding negative rate of change. Denote the half lines emanating from $\overline{\boldsymbol{x}}$ along the n_1 edges by

$$\zeta^j = \left\{ \boldsymbol{x} : \boldsymbol{x} = \overline{\boldsymbol{x}} - \lambda_j \boldsymbol{e}^j,\ \lambda_j \geq 0 \right\} \text{ for } j \in \mathcal{J}.$$

Taking each ζ^j one after the other, we can obtain a convex polyhedral cone containing $\boldsymbol{X}_0^i$. The fundamental idea of a cutting plane method for solving DBLP (1) is that along each ζ^j, we move a distance $\overline{\lambda}_j$ until we intersect a facet of $\boldsymbol{Y}_0(\alpha)$.

Denote by λ_j^+ the step-size along the positive extension of ζ^j, which is defined as

$$\lambda_j^+ = \sup\left\{\lambda_j : f(\overline{\boldsymbol{x}} - \lambda_j \boldsymbol{e}^j, \boldsymbol{y}) \geq \alpha \text{ for all } \boldsymbol{y} \in \boldsymbol{Y}_0\right\} \text{ for } j \in \mathcal{J}. \tag{2}$$

If $\lambda_j^+ = \infty$ for some $j \in \mathcal{J}$, we have to turn to the negative extension of ζ^j because $\forall \lambda_j > 0$, $\min_{\boldsymbol{y} \in Y_0} f(\overline{\boldsymbol{x}} - \lambda_j \boldsymbol{e}^j, \boldsymbol{y}) \geq \alpha$ along the positive extension of ζ^j, or equivalently, $\zeta^j \subset \boldsymbol{Y}_0(\alpha)$. Therefore, we intend to move as far as possible along the negative extension of ζ^j as long as we still lie in at least one half space $\mathcal{H}_i^+(\boldsymbol{x})$ defining $\boldsymbol{Y}_0(\alpha)$. As a result, λ_j^- is defined as

$$\lambda_j^- = \sup\left\{\lambda_j : f(\overline{\boldsymbol{x}} + \lambda_j \boldsymbol{e}^j, \boldsymbol{y}) \geq \alpha \text{ for some } \boldsymbol{y} \in \boldsymbol{Y}_0\right\} \text{ for } j \in \mathcal{J} \text{ and } \lambda_j^+ = \infty. \tag{3}$$

According to Theorem 4.1 in [13], define

$$\overline{\lambda}_j = \begin{cases} \lambda_j^+, \text{ if } 0 < \lambda_j^+ < \infty, \\ -\lambda_j^-, \text{ if } \lambda_j^+ = \infty \text{ and } 0 < \lambda_j^- < \infty. \end{cases} \tag{4}$$

Then, the inequality

$$\sum_{j \in \mathcal{J}} x_j / \overline{\lambda}_j \geq 1 \tag{5}$$

determines a valid polar cut being able to cut off $\overline{\boldsymbol{x}}$ without excluding any feasible point of $\boldsymbol{X}_0^i$ that yields a better objective value along with $\boldsymbol{y} \in \boldsymbol{Y}_0$. As (5) is organized in terms of $\mathcal{J}$, some appropriate linear transformation is necessary to represent it into the original variables.

During the derivation of $\overline{\lambda}_j$, we confront the following situations, several of which may incur computational issues.

(a). If $\lambda_j^+ = \infty$, $\forall j \in \mathcal{J}$, meaning that $\zeta^j \subset \boldsymbol{Y}_0(\alpha)$, $\forall j \in \mathcal{J}$, we can terminate the cutting plane method with the current best objective value α as the global minimum.

(b). If $0 < \lambda_j^+ < \infty$, we set $\overline{\lambda}_j = \lambda_j^+$ and continue to handle the next edge, ζ^{j+1}.

(c). If there exists some j such that $\lambda_j^+ = \infty$, we resort to the negative extension of ζ^j; see (3). Provided $0 < \lambda_j^- < \infty$, we can set $\overline{\lambda}_j = -\lambda_j^-$ by (4) and continue. Unfortunately, $\lambda_j^+ = \infty$ does not necessarily guarantee $0 < \lambda_j^- < \infty$; see **Example 1** for an illustration.

Example 1. *The example is taken from [19], in which* $\mathbf{c} = \boldsymbol{d} = (-1, -1)^t$,

$$\boldsymbol{C} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ \boldsymbol{A}_1 = \begin{pmatrix} 0 & 1 \\ -2 & -1 \\ 2 & -1 \end{pmatrix}, \ \boldsymbol{b}_1 = \begin{pmatrix} 2 \\ -2 \\ 2 \end{pmatrix}, \ \boldsymbol{A}_2 = \begin{pmatrix} -1 & 1 \\ 1 & 1 \\ 0 & -2 \end{pmatrix}, \ \boldsymbol{b}_2 = \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix}.$$

By **Algorithm 1**, *we can reach a PGM* $(0, 2, 2, 0)^t$ *with* $\alpha = -4$, $\overline{\boldsymbol{x}} = \overline{\boldsymbol{x}}^1 = (0, 2)^t$, *and its two adjacent vertices,* $\check{\boldsymbol{x}}_1 = (1, 0)^t$ *and* $\check{\boldsymbol{x}}_2 = (2, 2)^t$. *Since* $\overline{\boldsymbol{x}}$ *is non-degenerate, we have* $|\mathcal{J}| = 2$ *with* $\mathcal{J} = \{3, 4\}$. *Along the edge corresponding to the non-basic variable* x_3, *or equivalently, the edge determined by* $\overline{\boldsymbol{x}}$ *and* $\check{\boldsymbol{x}}_2$, *we first have* $\lambda_1^+ = \infty$ *with its objective as* -2. *Then, we turn to its negative extension and derive* $\lambda_1^- = \infty$ *with its objective as* -2 *again. Consequently, we reach a situation where* $\lambda_1^+ = \lambda_1^- = \infty$. *One possible approach is to set* $\overline{\lambda}_1 = \infty$ *so that the coefficient of its paired non-basic variable becomes zero. However, this approach can only take effect on a non-degenerate vertex, as with this example. Things become quite complicated for a degenerate vertex as it is hard to determine which non-basic variable to pair with the current edge.*

(d). If $\lambda_j^+ = 0$, we are obliged to discard the development of a polar cut and turn to a computationally expensive disjunctive cut since the edge under consideration leads to a degenerate pivot. Nevertheless, such a situation can be identified at an earlier stage by line 4, **Algorithm 1** when the algorithm intends to construct the neighborhood of $\tilde{\boldsymbol{x}}$. The treatment of degeneracy is out of the scope of this paper. Interested readers can refer to [1, 13, 20].

Apart from the derivation of $\overline{\lambda}_j$, (5) is only suitable for a non-degenerate vertex as each $\overline{\lambda}_j$ derived along ζ^j should be paired with one non-basic variable $x_j \in \mathcal{J}$. For a non-degenerate $\overline{\boldsymbol{x}}$, the associated simplex tableau is unique, and thereby ζ^j and the corresponding non-basic variable can be clearly identified. As for a degenerate $\overline{\boldsymbol{x}}$, however, such one-to-one correspondence vanishes since many simplex tableaux are associated with $\overline{\boldsymbol{x}}$, resulting in a lack of unified representation. This issue seems more or less related to (c) and (d), and we will propose a new method for a better treatment regardless of degeneracy in the next subsection.

3.2. Derivation of $\boldsymbol{e}^j$

Different from $\mathcal{J}$ defined in the last subsection, denote by $\tilde{\mathcal{J}} = \{1, 2, \ldots, |\tilde{\mathcal{J}}|\}$ the number of edges emanating from $\overline{\boldsymbol{x}}$. As a by-product of **Algorithm 1**, the information about all vertices adjacent to $\overline{\boldsymbol{x}}$ is already available during the derivation of a PGM. As a result, we can establish $\boldsymbol{e}^j$s along the positive extensions of ζ^js by

$$\boldsymbol{e}^j = \overline{\boldsymbol{x}} - \check{\boldsymbol{x}}_j, \ \forall \check{\boldsymbol{x}}_j \in \mathcal{N}_{\boldsymbol{X}_0^i}(\overline{\boldsymbol{x}}), \tag{6}$$

and thus enable the following computation to derive $\overline{\lambda}_j$s as usual. Note that this approach also works for a degenerate $\overline{\boldsymbol{x}}$ without imposing any additional computational workload. By doing so, we can avoid the complicated correspondence between $\overline{\lambda}_j$ (or ζ^j) and the non-basic variable in (5) given a degenerate $\overline{\boldsymbol{x}}$. Besides, given all the cutting points, a polar cut can be established without any further linear transformation, as has been done in the literature; see **Example 2**.

Example 2. *Take the example from [13], in which* $\boldsymbol{c} = (0, 0)^t$, $\boldsymbol{d} = (8, -6)^t$, $\boldsymbol{x} \geq \boldsymbol{0}$, $\boldsymbol{y} \geq \boldsymbol{0}$,

$$\boldsymbol{C} = \begin{pmatrix} 2 & -3 \\ -1 & 2 \end{pmatrix}, \ \boldsymbol{A}_1 = \begin{pmatrix} -2 & 5 \\ -3 & -2 \\ 0 & -1 \\ 3 & 2 \\ 2 & 12 \end{pmatrix}, \ \boldsymbol{b}_1 = \begin{pmatrix} 18 \\ -11 \\ -1 \\ 62 \\ 84 \end{pmatrix}, \ \boldsymbol{A}_2 = \begin{pmatrix} -1 & 1 \\ -3 & 4 \\ 4 & -5 \end{pmatrix}, \ \boldsymbol{b}_2 = \begin{pmatrix} -1 \\ -1 \\ 3 \end{pmatrix}.$$

By **Algorithm 1**, *we can first reach a PGM* $(1, 4, 2, 1)^t$ *with* $\overline{\boldsymbol{x}} = \overline{\boldsymbol{x}}^1 = (1, 4)^t$, $\alpha = 11$, *and* x_3 *and* x_4 *as the non-basic variables. By the simplex tableau corresponding to* $\overline{\boldsymbol{x}}$, *we have* $\boldsymbol{e}^3 = (-2/19, 3/19)^t$, *and thus* $\overline{\lambda}_3 = \lambda_3^+ = 95/3$. *As for* $\overline{\lambda}_4$, *with* $\boldsymbol{e}^4 = (-5/19, -2/19)^t$, *we first have* $\lambda_4^+ = \infty$ *and then derive* $\lambda_4^- = 494$ *along its negative extension.*

Nevertheless, as long as we have $\overline{\boldsymbol{x}} = (1, 4)^t$ *by* **Algorithm 1**, *the information about its two adjacent vertices,* $\check{\boldsymbol{x}}_1 = (3, 1)^t$ *and* $\check{\boldsymbol{x}}_2 = (6, 6)^t$, *is already available. Along the edge determined by* $\overline{\boldsymbol{x}}$ *and* $\check{\boldsymbol{x}}_1$, *we have* $\boldsymbol{e}^1 = \overline{\boldsymbol{x}} - \check{\boldsymbol{x}}_1 = (-2, 3)^t$, *with which we can derive* $\lambda_1^+ = 5/3$. *The corresponding cutting point is* $(13/3, -1)^t$ *with its objective as* 11. *Similarly,* $\boldsymbol{e}^2 = \overline{\boldsymbol{x}} - \check{\boldsymbol{x}}_2 = (-5, -2)^t$, $\lambda_2^- = -26$, *the cutting point is* $(-129, -48)$ *with its objective as* 11. *Both coincide with the previous results.*

Since $\overline{\boldsymbol{x}}$ is non-degenerate in **Example 2**, $|\mathcal{J}| = 2$ with $\mathcal{J} = \{3, 4\}$ corresponding to the indices of non-basic variables for the traditional method. For (6), $|\tilde{\mathcal{J}}| = 2$ just indicates the number of edges emanating from $\overline{\boldsymbol{x}}$, whereas $\tilde{\mathcal{J}} = \{1, 2\}$ represents the natural order of these two edges. The correspondence between indices and edges does not matter that much. Apparently, $\boldsymbol{e}^j$ established by (6) is more appealing for its accommodation to degeneracy. It does not require any additional computational effort such as the re-construction of the corresponding simplex tableau at $\overline{\boldsymbol{x}}$, or the transformation back to a representation of the original variables.

3.3. Derivation of λ_j^+

To derive λ_j^+, consider DBLP (1) and (2), in which we need to obtain

$$\begin{aligned}
&\max_{\lambda_j^+ \geq 0} \left\{ f\left(\overline{\boldsymbol{x}} - \lambda_j^+ \boldsymbol{e}^j, \boldsymbol{y}\right) \geq \alpha \text{ for all } \boldsymbol{y} \in \boldsymbol{Y}_0 \right\} \\
&= \max_{\lambda_j^+ \geq 0} \left\{ \min_{\boldsymbol{y} \in \boldsymbol{Y}_0} f\left(\overline{\boldsymbol{x}} - \lambda_j^+ \boldsymbol{e}^j, \boldsymbol{y}\right) \geq \alpha \right\} \\
&= \max_{\lambda_j^+ \geq 0} \left\{ \min_{\boldsymbol{y}} \left[\boldsymbol{c}^t \left(\overline{\boldsymbol{x}} - \lambda_j^+ \boldsymbol{e}^j\right) + \boldsymbol{d}^t \boldsymbol{y} + \left(\overline{\boldsymbol{x}} - \lambda_j^+ \boldsymbol{e}^j\right)^t \boldsymbol{C} \boldsymbol{y} \right] \geq \alpha \middle| \boldsymbol{A}_2 \boldsymbol{y} \leq \boldsymbol{b}_2, \boldsymbol{y} \geq \boldsymbol{0} \right\}.
\end{aligned} \tag{7}$$

By the LP duality theory, the foregoing can be rewritten as

$$\begin{aligned}
&\max_{\lambda_j^+ \geq 0} \left\{ \max_{\boldsymbol{u} \leq \boldsymbol{0}} \left[\boldsymbol{c}^t \left(\overline{\boldsymbol{x}} - \lambda_j^+ \boldsymbol{e}^j\right) + \boldsymbol{b}_2^t \boldsymbol{u} \right] \geq \alpha \middle| \left(\overline{\boldsymbol{x}} - \lambda_j^+ \boldsymbol{e}^j\right)^t \boldsymbol{C} + \boldsymbol{d}^t \geq \boldsymbol{u}^t \boldsymbol{A}_2 \right\} \\
&= \max_{\lambda_j^+ \geq 0} \left\{ \min_{\boldsymbol{u} \leq \boldsymbol{0}} \left(\boldsymbol{c}^t \boldsymbol{e}^j \lambda_j^+ - \boldsymbol{b}_2^t \boldsymbol{u} \right) \leq \boldsymbol{c}^t \overline{\boldsymbol{x}} - \alpha \middle| \boldsymbol{C}^t \boldsymbol{e}^j \lambda_j^+ + \boldsymbol{A}_2^t \boldsymbol{u} \leq \boldsymbol{C}^t \overline{\boldsymbol{x}} + \boldsymbol{d} \right\},
\end{aligned} \tag{8}$$

which is equivalent to finding the maximal λ_j^+ such that the following linear system is feasible

$$\begin{aligned}
&\max \quad \lambda_j^+ \\
&\text{s.t.} \quad \begin{pmatrix} \boldsymbol{c}^t \boldsymbol{e}^j & -\boldsymbol{b}_2^t \\ \boldsymbol{C}^t \boldsymbol{e}^j & \boldsymbol{A}_2^t \end{pmatrix} \begin{pmatrix} \lambda_j^+ \\ \boldsymbol{u} \end{pmatrix} \leq \begin{pmatrix} \boldsymbol{c}^t \overline{\boldsymbol{x}} - \alpha \\ \boldsymbol{C}^t \overline{\boldsymbol{x}} + \boldsymbol{d} \end{pmatrix}, \quad \lambda_j^+ \geq 0, \quad \boldsymbol{u} \leq \boldsymbol{0}.
\end{aligned} \tag{9}$$

So, unlike the modified Newton's method, we can obtain λ_j^+ by solving only one LP. Unfortunately, this technique cannot be applied to the derivation of λ_j^- since the equivalence between (8) and (9) cannot be established. Thus, the modified Newton's method should be employed.

Example 3. *Still take* **Example 2** *as an illustrative example. By* **Algorithm 1***, we are able to reach a PGM* $(1,4,2,1)^t$ *with* $\overline{\boldsymbol{x}} = (1,4)^t$*,* $\alpha = 11$*, and* x_3 *and* x_4 *as non-basic variables. Using* $\boldsymbol{e}^3 = (-2/19, 3/19)^t$ *and system (9), we have* $\overline{\lambda}_3 = \lambda_3^+ = 95/3$*, as expected.*

3.4. Derivation of $\overline{\lambda}_j$

Algorithm 2 describes the derivation of $\overline{\lambda}_j$ for $j \in \tilde{\mathcal{J}}$. In comparison with the classical cutting plane method in [13], the following aspects need to be clarified.

(a) The loop from line 1 to 3 to derive λ_j^+, $\forall j \in \tilde{\mathcal{J}}$ is reasonable because it may cause an early stop and does not sacrifice any additional computational effort. Suppose $\lambda_j^+ = \infty$, $\forall j \in \tilde{\mathcal{J}}$, meaning that we have already solved the problem. Then, all computational effort spent in deriving λ_j^-s is unnecessary if we turn to λ_j^- as soon as we identify $\lambda_j^+ = \infty$, as was done in [13]. Even though this is not the case, the computational cost on λ_j^+s is inevitable for either $\lambda_j^+ < \infty$ or λ_j^- given $\lambda_j^+ = \infty$.

(b) If either λ_j^+ or λ_j^- is finite, i.e., line 9 and 15, we can derive $\overline{\lambda}_j$ by (4). However, in line 13, we are confronted with the case where both $\lambda_j^+ = \infty$ and $\lambda_j^- = \infty$. If $\overline{\boldsymbol{x}}$ is non-degenerate, we can set $\overline{\lambda}_j = \infty$, or equivalently, set the coefficient of paired non-basic variable to zero. As for a degenerate $\overline{\boldsymbol{x}}$, we simply set $\overline{\lambda}_j = 2$ so that at least the adjacent vertex along this edge can be cut off, just like what a face cut does. In fact, any value such that $\overline{\lambda}_j > 1$ can realize this.

Algorithm 2: Derivation of $\overline{\lambda}_j$s

```
Input: $\overline{\boldsymbol{x}}$, $\boldsymbol{Y}_0$, $\boldsymbol{e}^j$s, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{C}$, $\alpha$, $\tilde{\mathcal{J}}$
Output: $\overline{\lambda}_j$s
1  foreach $j \in \tilde{\mathcal{J}}$ do
2      compute $\lambda_j^+$ by system (9);
3  end
4  if $\lambda_j^+ = \infty,\ \forall j \in \tilde{\mathcal{J}}$ then
5      terminate with $\alpha$ as the global minimum;
6  else
7      foreach $j \in \tilde{\mathcal{J}}$ do
8          if $\lambda_j^+ < \infty$ then
9              $\overline{\lambda}_j \leftarrow \lambda_j^+$;
10         else
11             $\lambda_j^- \leftarrow \infty$ and $\mathbb{O} \leftarrow \max_{\boldsymbol{y}\in\boldsymbol{Y}_0} f\left(\overline{\boldsymbol{x}} + \lambda_j^- \boldsymbol{e}^j, \boldsymbol{y}\right)$;
12             if $\mathbb{O} \geq \alpha$ then
13                 $\overline{\lambda}_j \leftarrow \infty$ for a non-degenerate $\overline{\boldsymbol{x}}$, or $\overline{\lambda}_j \leftarrow 2$ for a degenerate $\overline{\boldsymbol{x}}$;
14             else
15                 $\overline{\lambda}_j \leftarrow -\lambda_j^-$ by the modified Newton's method;
16             end
17         end
18     end
19 end
```

(c) By [13], we first need to compute $\mathbb{O}' = \min_{\boldsymbol{y}\in\boldsymbol{Y}_0} f\left(\overline{\boldsymbol{x}} - \lambda_j^+ \boldsymbol{e}^j, \boldsymbol{y}\right)$ with one LP to determine whether it is necessary to compute λ_j^-. If $\mathbb{O}' < \alpha$, the modified Newton's method should be carried out for λ_j^+, demanding some more LPs. By contrast, **Algorithm 2** can derive λ_j^+ with only one LP by (9). The advantages are twofold. By one LP, **Algorithm 2** can determine not only λ_j^+ if $\lambda_j^+ < \infty$, but also the necessity for computing λ_j^- if $\lambda_j^+ = \infty$.

(d) If $\overline{\boldsymbol{x}}$ is non-degenerate, we can simply take $\tilde{\mathcal{J}} = \mathcal{J}$ and carry out **Algorithm 2**.

(e) By [13], $\lambda_j^+ = 0$ indicates that the edge under consideration results in a degenerate pivot, which, in turn, leads to the development of a computationally expensive disjunctive cut. The wasted effort depends on when $\lambda_j^+ = 0$ occurs. Suppose for $j_1, j_2, \ldots, j_t \in \tilde{\mathcal{J}}$, $\lambda_j^+ = 0$ does not take place until we intend to identify $\overline{\lambda}_{j_t}$. This incurs a waste of all the previous computational effort spent in deriving $\overline{\lambda}_{j_1}, \overline{\lambda}_{j_2}, \ldots, \overline{\lambda}_{j_{t-1}}$ before we develop a disjunctive cut from scratch. Nevertheless, in line 4, **Algorithm 1**, prior to the evaluation of objectives regarding the vertices adjacent to $\widetilde{\boldsymbol{x}}$, its degeneracy can be readily determined without any additional computation. Knowing this enables an early choice of the correct approach to the development of a valid cut, and thereby save the computational effort.

(f) The modified Newton's method may incur many LPs for a piecewise linear and concave function with breakpoints occurring whenever the vertex solution optimizing this function changes. In Figure 1, we use the notations as in [13] for illustration. Figure 1(a) indicates that it may be possible to skip the workload over line segment $\boldsymbol{FG}$ when deriving λ^{i+2} from λ^{i+1} due to the relationship among the slopes of different pieces. Things can become even better if a single derivation of some λ^i can ignore the workload over several pieces. Nevertheless, we can also confront the case where workload over no line segments can be neglected; see Figure 1(b). As an LP should be carried out in search of the optimizer in $\boldsymbol{Y_0}$

for each piece when we intend to make progress from λ^i to λ^{i+1}, many LPs are required. It is this aspect that (9) can take effect in the derivation of λ_j^+.

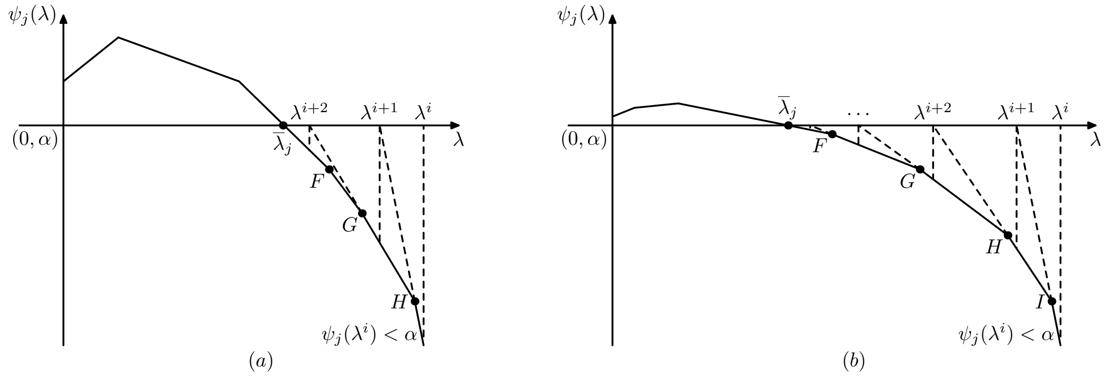

Figure 1: *The Modified Newton's Method.*

4. Numerical Results

To evaluate the performance of two methods, we carry out the numerical experiments on a PC equipped with Intel(R) Core(TM) i5-6267U CPU @ 2.90GHz and 4G memory. Gurobi 11.0.0 serves as the LP solver. We randomly generate 32 test instances by [19] for each dimension and average the corresponding results. In addition, we impose a limit of 300 seconds over the total computing time for each test instance, beyond which the instance is considered unsolvable.

As the dual method can only take effect on positive cutting points, we evaluate their performance on those instances whose percentages of positive cutting points in a test instance exceed 40%, 50%, and 60%, respectively. For a specific instance, we compute its percentage as

$$\text{percentage} = \frac{\text{the number of positive cutting points}}{\text{the dimension of an instance}}, \tag{10}$$

and we only consider qualified instances for each dimension. Apparently, beyond 60%, positive cutting points are dominating. By contrast, below 40%, the dual method can hardly influence the computational performance due to the large number of negative cutting points.

Figure 2 illustrates the performance between two methods with respect to three thresholds imposed on percentages. The computing time along three y axes includes the average CPU time to derive both positive and negative cutting points for the qualified instances with dimension $\boldsymbol{n}$. For a clear presentation, we deliberately fix the scale on each y-axis from 0 to 225 seconds. By the left sub-figure where the threshold is 40%, the dual method performs inferior to the modified Newton's method for most dimensions. When it comes to 50%, on average, the performance of two methods overlaps so that they become competitive; see the middle sub-figure. Finally, as we further increase the threshold to 60%, the dual method runs faster for almost all dimensions and the computing time savings gradually improve. This seems reasonable since the higher the percentage of positive cutting points, the more computing time we are able to save by utilizing the dual method. However, we cannot exclude the possibility that for an extreme instance with even a dominating number of positive cutting points, the performance of the dual method may still appear inferior due to the formulation of a not easy-to-solve LP program by (9). We can safely conclude that, in general, along with an increase in the percentage of positive cutting points, the performance of the dual method can improve little by little. This again verifies the influence the dual method will have on the derivation of positive cutting points.

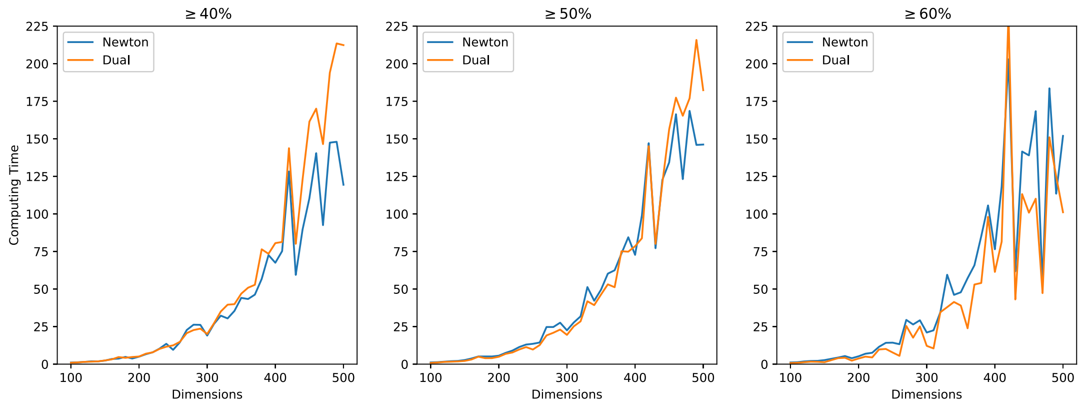

Figure 2: *Performance between two Methods.*

Table 1 provides more detailed results but with a step-size of 20. The results are concerned with the savings in computing time, which are calculated according to

$$\text{time\_savings} = \frac{\text{modified Newton's computing time - dual's computing time}}{\text{modified Newton's computing time}} \times 100\%. \quad (11)$$

n \ %	$\geq 20\%$	$\geq 30\%$	$\geq 40\%$	$\geq 50\%$	$\geq 60\%$
$n = 100$	3.098	6.118	13.545	30.233	34.609
$n = 120$	-8.132	-5.644	-2.487	9.413	27.831
$n = 140$	-6.410	0.372	0.235	15.237	29.131
$n = 160$	-1.483	7.132	7.352	17.422	24.736
$n = 180$	5.423	5.842	12.762	22.125	21.014
$n = 200$	-15.731	-12.484	-2.065	12.495	28.177
$n = 220$	-4.362	-4.578	0.244	13.960	40.838
$n = 240$	5.838	7.271	14.375	12.858	28.794
$n = 260$	-15.602	-18.228	-1.996	11.893	58.458
$n = 280$	-0.011	-5.297	13.603	15.472	33.331
$n = 300$	-9.586	-1.770	-5.498	13.563	42.222
$n = 320$	-2.787	-13.891	-8.657	10.408	0.039
$n = 340$	-5.911	-7.169	-12.717	6.885	10.250
$n = 360$	-20.376	-16.299	-17.347	11.837	58.278
$n = 380$	-37.716	-44.139	-34.829	-2.073	36.517
$n = 400$	-20.372	-22.555	-19.375	-7.910	19.709
$n = 420$	-43.160	-38.215	-12.061	1.357	-14.964
$n = 440$	-28.739	-29.949	-37.160	1.804	20.025
$n = 460$	-39.937	-31.819	-21.101	-6.634	34.616
$n = 480$	-40.161	-40.161	-31.578	-4.978	17.754
$n = 500$	-78.464	-79.384	-77.831	-24.743	33.452

Table 1: *Savings in Average Computing Time (%).*

In order to perceive the effectiveness of the dual method, we impose two more thresholds on percentages, i.e., 20% and 30%. It can be observed that for the first three thresholds (column 2–4), most savings in computing time are negative. By (11), it means that the performance of the dual method, in most cases, is inferior to that of the modified Newton's method. Nevertheless, as the threshold is increased to 50%, negative percentages become much less. When it comes to 60%, there is only one negative percentage left. For large-scale instances where $\boldsymbol{n} = 480$ or 500, the improvement in computational performance brought by the dual method appears more remarkable. Together with Figure 2, the improvement in percentages of savings in computing time for low dimensional instances is less substantial than that for high dimensional instances. For example, the same improvement when $\boldsymbol{n} = 100$ and $\boldsymbol{n} = 500$ can lead to great difference in their respective absolute improvements in the computing time.

Table 2 shows the number of LPs that will be performed by the modified Newton's method and the dual method, respectively. It gives the percentages of savings in the number of LPs according to

$$\text{LP\_savings} = \frac{\text{Newton's LPs} - \text{Dual's LPs}}{\text{Newton's LPs}} \times 100\% \tag{12}$$

$\boldsymbol{n}$ \ %		$\geq 20\%$	$\geq 30\%$	$\geq 40\%$	$\geq 50\%$	$\geq 60\%$
$\boldsymbol{n} = 100$	modified Newton's method	268	265	263	260	262
	dual method	204	197	186	165	160
	savings	**23.7%**	**25.7%**	**29.4%**	**36.6%**	**39.1%**
$\boldsymbol{n} = 150$	modified Newton's method	394	381	379	372	368
	dual method	289	268	262	246	230
	savings	**26.6%**	**29.7%**	**31.0%**	**33.9%**	**37.6%**
$\boldsymbol{n} = 200$	modified Newton's method	518	513	502	492	489
	dual method	372	362	345	328	304
	savings	**28.1%**	**29.5%**	**31.2%**	**33.4%**	**37.9%**
$\boldsymbol{n} = 250$	modified Newton's method	681	680	671	650	667
	dual method	499	488	449	402	382
	savings	**26.7%**	**28.2%**	**33.2%**	**38.2%**	**42.8%**
$\boldsymbol{n} = 300$	modified Newton's method	821	809	775	767	769
	dual method	617	586	534	494	477
	savings	**24.9%**	**27.5%**	**31.1%**	**35.6%**	**37.9%**
$\boldsymbol{n} = 350$	modified Newton's method	989	982	975	960	948
	dual method	726	705	685	637	568
	savings	**26.6%**	**28.2%**	**29.7%**	**33.6%**	**40.0%**
$\boldsymbol{n} = 400$	modified Newton's method	1155	1154	1143	1123	1104
	dual method	810	788	740	659	622
	savings	**29.9%**	**31.7%**	**35.3%**	**41.3%**	**43.6%**
$\boldsymbol{n} = 450$	modified Newton's method	1348	1355	1341	1333	1353
	dual method	922	910	858	784	757
	savings	**31.6%**	**32.8%**	**36.0%**	**41.2%**	**44.0%**
$\boldsymbol{n} = 500$	modified Newton's method	1532	1480	1455	1429	1481
	dual method	1090	996	949	878	834
	savings	**28.9%**	**32.7%**	**34.8%**	**38.6%**	**43.7%**

Table 2: *Savings in the Number of LPs (%).*

· As has been expected, the percentages of savings in the number of performed LPs gradually increase while the thresholds rise from 20% to 60%; see rows in bold texts. Nevertheless, such a trend does not necessarily imply a similar tendency in computing time. By (9), although we are able to re-formulate the problem to derive λ_j^+ as a single LP, its dimension and constraints are both greater than one LP in the modified Newton's method. By Table 2, the average number of LPs in the modified Newton's method to derive $\overline{\lambda}_j$ lie between 2 to 3 times. Therefore, it is not surprising to find out that the computing time of the dual method is more than solving one LP in the modified Newton's method, but less than solving 3 LPs. This is essential for the dual method to run faster when the thresholds increase. Besides, since the difficulty levels for solving

LPs varies, it may cost more computing time than expected when the dual method is faced with a troubling LP, and vice versa. That is why we average the results of many instances. Given an enhancement to the ability of an LP solver to solve a single large-scale LP, the computational performance of the dual method can be further improved.

5. Conclusion

In this paper, we develop a new method for the derivation of positive cutting points based on the LP duality theory, and a new technique for the derivation of $\boldsymbol{e}^j$s as an accommodation to degeneracy without any additional computational workload. Numerical results demonstrate that along with a rise in the percentage of positive cutting points, the computational performance for the generation of a polar cut can, on average, be improved by utilizing the dual method.

Acknowledgements

The research was partially supported by the Fundamental Research Funds for the Central Universities (2022TD001, 2023JJ002, 2023JX018), Young Faculty Research Fund of BFSU (2019-ZYQNJS-002), and NSFC (71801015).

References

[1] Chen, X., Zhang, J., Ding, X., Yang, T. and Qian, J. (2019). Location of a conservative hyperplane for cutting plane methods in disjoint bilinear programming. Optimization Letters, 13(7), 1677–1692. doi: 10.1007/s11590-018-01382-w

[2] Gadhi, N.A. and Lafhim, L. (2019). Optimality conditions for a bilevel optimization problem in terms of KKT multipliers and convexificators. Croatian Operational Research Review, 10(2), 329–335. doi: 10.17535/crorr.2019.0026

[3] Gallo, G. and Ülkücü, A. (1977). Bilinear programming: An exact algorithm. Mathematical Programming, 12(1), 173–194. doi: https://doi.org/10.1007/BF01593787

[4] Kheirfam, B. (2019). A new full-NT step interior-point method for circular cone optimization. Croatian Operational Research Review, 10(2), 275–287. doi: 10.17535/crorr.2019.0023

[5] Kheirfam, B. (2023). A new search direction for full-Newton step infeasible interior-point method in linear optimization. Croatian Operational Research Review, 14(2), 193–202. doi: 10.17535/crorr.2023.0016

[6] Konno, H. (1976). A cutting plane algorithm for solving bilinear programs. Mathematical Programming, 11(1), 14–27. doi: 10.1007/BF01580367

[7] Meyer, C. (2000). A simple finite cone covering algorithm for concave minimization. Journal of Global Optimization, 18(4), 357–365. doi: 10.1023/A:1026548217241

[8] Porembski, M. (1999). How to extend the concept of convexity cuts to derive deeper cutting planes. Journal of Global Optimization, 15(4), 371–404. doi: 10.1023/A:1008315229750

[9] Porembski, M. (2001). Finitely convergent cutting planes for concave minimization. Journal of Global Optimization, 20(2), 113–136. doi: 10.1023/A:1011240309783

[10] Porembski, M. (2004a). Cutting planes for low-rank-like concave minimization problems. Operations Research, 52(6), 942–953. doi: 10.1287/opre.1040.0151

[11] Porembski, M. (2004b). A new successive partition algorithm for concave minimization based on cone decomposition and decomposition cuts. Journal of Global Optimization, 29(2), 191–224. doi: 10.1023/B:JOGO.0000042113.68493.da

[12] Porembski, M. (2008). On the hierarchy of γ-valid cuts in global optimization. Naval Research Logistics, 55(1), 1–15. doi: 10.1002/nav.20257

[13] Sherali, H. and Shetty, C. (1980). A finitely convergent algorithm for bilinear programming problems using polar cuts and disjunctive face cuts. Mathematical Programming, 19(1), 14–31. doi: 10.1007/BF01581626

[14] Singh, V. P. and Chakraborty, D. (2019). Bi-level optimization based on fuzzy if-then rule. Croatian Operational Research Review, 10(2), 315–328. doi: 10.17535/crorr.2019.0025

[15] Tuy, H. (1964). Concave programming under linear constraints. Soviet Maththematics, 5, 1437–1440.

[16] Vaish, H. (1974). Nonconvex Programming with Applications to Production and Location Problems. PhD thesis, Georgia Institute of Technology.

[17] Vaish, H. and Shetty, C. (1976). The bilinear programming problem. Naval Research Logistics, 23(2), 303–309. doi: 10.1002/nav.3800230212

[18] Vaish, H. and Shetty, C. (1977). A cutting plane algorithm for the bilinear programming problem. Naval Research Logistics, 24(1), 83–94. doi: 10.1002/nav.3800240107

[19] Vicente, L., Calamai, P. and Júdice, J. (1992). Generation of disjointly constrained bilinear programming test problems. Computational Optimization and Applications, 1(3), 299–306. doi: 10.1007/BF00249639

[20] Zhang, J., Chen, X. and Ding, X. (2017). Degeneracy removal in cutting plane methods for disjoint bilinear programming. Optimization Letters, 11(3), 483–495. doi: 10.1007/s11590-016-1016-6

[21] Zwart, P. (1973). Nonlinear programming: Counterexamples to two global optimization algorithms. Operations Research, 21(6), 1260–1266. doi: 10.1287/opre.21.6.1260