INFORMATICA, 2025, Vol. 36, No. 4, 765–795

DOI: https://doi.org/10.15388/25-INFOR609

Polytopal Spatial Branch and Bound Global Optimization Algorithm ☆

L.G. CASADO[1,*], B. G.-TÓTH[2], E.M.T. HENDRIX[3], F. MESSINE[4]

[1] *University of Almería, Almería, CEINSA, Spain*
[2] *University of Szeged, Szeged, Hungary*
[3] *Universidad de Málaga, Málaga, Spain*
[4] *University of Toulouse, LAPLACE-ENSEEIHT, Toulouse, France*
e-mail: leo@ual.es, boglarka@inf.szte.hu, eligius@uma.es, messine@laplace.univ-tlse.fr

Received: September 2025; accepted: October 2025

Abstract. Spatial Global Optimization branch and bound (B&B) methods aim at enclosing global minimum points in a guaranteed way with a certain accuracy. We extend simplicial B&B (sBB) concepts to polytopal B&B (pBB), with polytope subsets. The main challenges are: polytope division and extension of monotonicity tests theoretically and algorithmically. We compare the performance of interval B&B with linear constraints (iBBLC), sBB and pBB algorithms, to determine the most efficient B&B algorithm for different types of instances.

Key words: polytope, branch and bound, monotonicity, interval arithmetic, global optimization.

1. Introduction

The research question dealt with in this paper is whether a spatial branch and bound method can be built using polytopal subsets in order to solve linearly constrained global optimization problems. How can division of polytopes be done efficiently? What information should be stored about a polytope subset? How can monotonicity considerations be used effectively?

Interval Arithmetic based branch and bound has a long tradition in monotonicity considerations to remove subsets, see Hansen and Walster (2004), Kearfott (1992). In our investigation, we consider Interval Arithmetic as a relative easy way to get bounds of the objective function and its derivatives over an n-dimensional interval vector, a so-called box. Interval Arithmetic spatial branch and bound (iBB) is a well established concept elaborated in codes with various acceleration methods. The idea is that one can remove interior boxes where the function is monotonic over the box and reduce the dimension with respect to a monotonic component when a box facet is at the boundary of the search space. Simplicial branch and bound (sBB), see Paulavičius and Žilinskas (2014), initially

☆This work has been funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe", Grant PID2021-123278OB-I00 of the Spanish ministry of Science and Innovation.
*Corresponding author.

considered full dimensional subsets. We extended the monotonicity concepts of iBB to exploit derivative bounds of the interval hull of a convex set to construct monotonicity tests in sBB, see Casado *et al.* (2022, 2025). Specifically, we showed that one can reduce to lower dimensional faces of the subset that are certified to contain all possible minima of the subset.

Recent numerical experiments comparing iBB with linear constraints (iBBLC) with sBB show that sBB may provide advantage in computation time. This advantage is clear for non full-dimensional feasible sets which are not parallel to the axes, see G.-Tóth *et al.* (2024). We found that this is mainly due to not having to handle the linear constraints in sBB, as simple linear constraints actually define a simplex or a polytope. Thus, our research question addresses the potential advantages of using a polytopal B&B (pBB) method rather than iBBLC and sBB approaches to solve linearly constrained global optimization problems.

The objective of a B&B algorithm is to enclose all global minimum points with the minimum value of

$$\min_{x \in q} h(x), \tag{1}$$

where in our case feasible set q is a polytope, while $h(x)$ is a continuously differentiable objective function.

Algorithm 1 Spatial BB (h, q, α)

Require: h: objective function, q: feasible set, α: termination accuracy

```
Evaluate bounds on h(q)                                   ▷ Bounding rule
Λ ← q                                                     ▷ Store q in working list Λ
while not termination criterion do                        ▷ Termination rule: based on α
    S ← Λ                                                 ▷ Selection rule: extract a set S from Λ
    if S cannot be rejected then                          ▷ Rejection rules
        if S can be reduced then
            Generate lower dimensional subsets S_j        ▷ Reduction rules
        else                                              ▷ Not rejected neither reduced
            Generate same dimensional subsets S_j         ▷ Division rule
        for generated subsets S_j do
            Evaluate bounds on h(S_j)                     ▷ Bounding rule
            Λ ← S_j                                       ▷ Store S_j in Λ
```

Algorithm 1 shows a generic scheme of a spatial branch and bound algorithm. B&B algorithms can be characterized by a set of rules. Although we focus here on the polytopal division, rejection and reduction rules, other branch and bound rules are bounding, selection and termination. They all play a role in the efficiency of the algorithm. In the presented pseudocode of Algorithm 1, rejection and reduction tests are done after extracting

a set from the working list Λ. In practice, performing those tests before storing a subset into Λ reduces the size of Λ and its management cost.

Our initial hypotheses, which support the idea that a polytopal B&B can be useful, are as follows:

1. Bisection of the widest component of the interval hull of a polytope in its division generates cuts parallel to the axes. This may lead to tighter enclosure of a resulting subset p from a division of a polytope. This enclosure is given by the interval hull of p: $\square p$. The motivation is to get tighter bounds of $h(p)$ and $\nabla h(p)$ obtained by Interval Arithmetic with Automatic Differentiation over $\square p$, in comparison with other divisions, as the radial division.
2. By using a convex and closed feasible polytopal set in the pBB algorithm, we get rid of the linear constraints (LCs) used by an iBBLC algorithm. This has several advantages:
   a) We do not have to deal with feasibility.
   b) Extension of sBB concepts on monotonicity to pBB provides reduction of dimensionality of a polytopal subset.
   c) We can avoid generating huge lists of boxes around boundary minima in iBBLC.

Development of a pBB algorithm provides several challenges. The following list summarizes the main contributions of this study:

- The use of a polytope description based on vertices, edges and facets in a pBB algorithm. We avoid the use of LP solvers or matrix computation. So, we do not have to deal with their precision issues in the pBB algorithm.
- Extension of theoretical sBB results about monotonicity rejection and reduction to pBB.
- The development of coordinate-wise bisection of a polytope defined by vertices, edges and facets. A challenge is to determine the edges of a large dimensional cutting facet.
- A way to keep track of the border status of the facets of generated polytopes by division and how to update the border status of the edges and facets after a reduction.
- The efficiency comparison of iBBLC, sBB and pBB algorithms over a set of designed instances to characterize the cases in which pBB may provide an advantage.

Due to algorithm comparison, we use similar branch and bound rules (see Algorithm 1) and basic accelerating tools, up to the first derivative for iBBLC, sBB and pBB. More advanced acceleration tools have not been developed for neither sBB nor pBB algorithms yet, see Fernández and G.-Tóth (2022), Gencsi and G.-Tóth (2025).

To address the above contributions, Section 2 introduces mathematical notation and Section 3 provides properties of monotonicity over polytope sets. Section 4 describes a specifically designed algorithm. Section 5 compares the performance of the diverse algorithms, while Section 6 summarises our findings.

2. Mathematical Notations

We consider the minimization of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, over a polytope as feasible set. The 'polytope', as a convex hull of a finite number of points,

is also called convex polytope in literature. Our main motivation is the fact that many nonlinear problems have linear constraints together with bound constraints. This implies having a polytope as feasible set. The dimension of such polytope can be less than n, i.e. the feasible set is in a lower dimensional space than that of the objective function h. The exact dimension of a polytope $q \subset \mathbb{R}^n$ is theoretically defined as the dimension of its affine hull: $\dim(q) := \dim(\mathrm{aff}(q))$. Notice that also the concepts of relative interior $\mathrm{rint}(q)$ and relative boundary apply.

Formally, let a polytope $q := \mathrm{conv}(W)$ be defined as the convex hull of its vertex set W, that is, a set of $|W|$ vectors $W := \{w_1, \ldots, w_{|W|}\} \subset \mathbb{R}^n$. We assume the vertices to be extreme points of q. In this context, consider the dimension of a polytope as the dimension of its affine hull, which can be computed from

$$\dim(q) = \mathrm{Rank}\big(\{w_2 - w_1, \ldots, w_{|W|} - w_1\}\big).$$

Definition 1. A polytope q with dimension $m = \dim(q)$ is called an *m-polytope*.

Therefore, a segment is a 1-polytope in 1-dimensional space or higher, a triangle and a square are 2-polytopes in 2-dimensional space or higher and so on. We will also use the corresponding notation of an $(m-1)$-facet or k-face, $k < m-1$.

Our experiments in G.-Tóth *et al.* (2024) with iBBLC and sBB codes taught us that there exist cases where using partition sets based on simplices is more effective than iBBLC when the feasible set dimension is lower than n, whereas partition sets based on boxes can exploit better the acceleration devices developed for them when in fact $\dim(q) = n$ and the global minimum is in the interior of q. Our main research question is whether we could simply use polytopal partition sets in a direct way. How can they be subdivided efficiently and how can monotonicity over them be exploited?

In this concept, the feasible set q and also any partition set (subset) p is a polytope $p := \mathrm{conv}(V) \subseteq q = \mathrm{conv}(W)$, where $V := \{v_1, \ldots, v_{|V|}\} \subset \mathbb{R}^n$ is a set of extreme points of p. We can use λ coordinates to describe partition set p

$$p = \left\{ x = \sum_{j=1}^{|V|} \lambda_j v_j \;\middle|\; \lambda \geqslant 0, \sum_{j=1}^{|V|} \lambda_j = 1 \right\} \tag{2}$$

and its relative interior

$$\mathrm{rint}(p) = \left\{ x = \sum_{j=1}^{|V|} \lambda_j v_j \;\middle|\; \lambda > 0, \sum_{j=1}^{|V|} \lambda_j = 1 \right\}. \tag{3}$$

The relative boundary of a polytope is the difference $\partial p = p \setminus \mathrm{rint}(p)$. Notice that the relative boundary cannot be written concisely with the λ coordinates, when p is not a simplex, as having some 0 in λ does not necessarily mean that the point is at the boundary.

The interval hull of p is given by $\square p := [\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$, where $\underline{\boldsymbol{x}}_i := \min_{v_j \in V} v_{ji}$ and $\overline{\boldsymbol{x}}_i := \max_{v_j \in V} v_{ji}$, where each vertex $v_j = (v_{j1}, \ldots, v_{jn})^T \subset \mathbb{R}^n$. Table 1 summarizes several symbols in this paper.

Table 1
Used symbols.

n	objective function dimension, number of components in vertices and points.
q	polytope feasible set $q = \text{conv}(W)$ with vertex set W.
p	polytope subset $p \subseteq q$, $p = \text{conv}(V)$ with vertex set V.
m	dimension of a polytope subset p for an m-polytope.
$\square p$	interval hull of subset p.
$V \subset \mathbb{R}^n$	vertex set $\{v_1, \ldots, v_{\lvert V\rvert}\}$.
$E \subset \mathbb{R}^n$	edge set $\{e_1, \ldots, e_{\lvert E\rvert}\}$, $e = (u, v)$, $u, v \in V$.
$F \subset \mathbb{R}^n$	facet set $\{f_1, \ldots, f_{\lvert F\rvert}\}$, $f = \{e_i, \ldots, e_j \in E\}$, of m-polytope p.
$d \subset \mathbb{R}^n$	directional vector as difference $d = y - x$ with $x, y \in p$.

For any polytope q, the minimum is attained either at one of its faces, i.e. in a vertex, edge, higher dimensional face; or in the relative interior of q. The advantage of a polytopal branch and bound algorithm is that the feasible set is exactly covered by the partition sets, thus feasibility problems cannot occur. In iBBLC, the undetermined partition overestimates the feasible volume. This means that if the feasible m-set, $m > 0$, is lower in dimension than the objective function ($m < n$), not being parallel to the axes, it is certainly overestimated by any non-degenerate n-box in an iBBLC algorithm.

An sBB may start with an exact simplicial partition of the polytope. However, this requires finding a simplicial partition of the polytope, and as we will show, it might be less efficient than dealing directly with the feasible set as a polytope. In earlier studies (Casado *et al.*, 2022; G.-Tóth *et al.*, 2024), we realised that the concept of monotonicity over a subset $p \subseteq q$ is relevant in rejection and reduction rules with respect to the border status of its facets. We discuss this aspect in the next section.

3. Theoretical Aspects

In this section, we discuss monotonicity considerations and define the related border status of facets. The relation is that when the objective function is monotone over a polytope $p \subseteq q$, the optimum can only be attained at its boundary. If that boundary is in the relative interior of q, p can be discarded from the search for the global minimum. This is the reason to keep track of the border status of the facets of p.

3.1. *Border Status of Facets in Polytopes*

There are various notions about being at the boundary of the feasible set. The most general case is to say that a polytope p is at the boundary of q, if $p \subseteq \partial q$. Now, in our case we want a subset to be in the relative boundary of the feasible set. To make it clear, we introduce the notion of border as follows.

Definition 2. Given feasible area q. An m-polytope p is called *border* with respect to q if there exists an m-polytope face φ of q, such that $p \subseteq \varphi$.

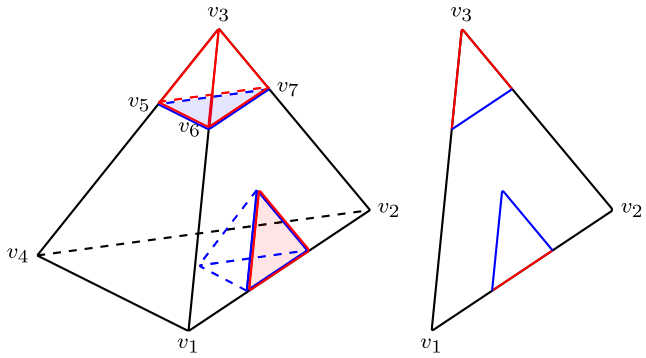

Fig. 1. In the left graph, q is a 3-simplex with two 3-sub-simplices in it. Consider the 2-facets of the sub-simplices with red edges; they are border, whereas those with blue edges are not border. Facet (v_3, v_6, v_7) is border, but facet (v_5, v_6, v_7) is not. The status depends on the face being a subset of a 2-facet φ of q. In the right graph, consider 2-facet $f_1 = (v_1, v_2, v_3)$ of q. Edges (1-simplex) in red of 2-simplices are border and those in blue are not. The status depends on the existence of a 1-face (edge) φ of q that encloses them. Notice that all coloured edges in the right graph are on the boundary of q, but the blue ones are not on the relative boundary of f_1.

Figure 1 illustrates when facets are border or not. We extend the concept of border status of a face defining its border level.

Definition 3. Given feasible area q, an m-polytope subset p and an $(m-1)$-polytope facet f of p. The border level $\boldsymbol{bl}(f)$ is defined by the dimension of the minimum dimensional face φ of q, such that $f \subseteq \varphi$.

The practical impact of certifying the border level of polytope subsets is given in the following results, which follow directly from the definitions.

Lemma 1. *Given feasible area q, an m-polytope subset p and an $(m-1)$-polytope facet f of p. Facet f is on the relative boundary of q if its border level $\boldsymbol{bl}(f) = m-1$, and not on the boundary if $\boldsymbol{bl}(f) = m$.*

Lemma 2. *Given feasible area q and m-polytope subset p. A border k-face $f \subset p$ has border level $\boldsymbol{bl}(f) = k$, $k < m$. A non-border k-face $f \subset p$ has border level $\boldsymbol{bl}(f) > k$.*

Notice that in order to check the border status of a face f using the above definition and lemmas, is complicated. It requires the set of faces φ of q stored with the dimension and a check whether one of them contains face f. For an n-box, the faces are easily defined by lower and upper bounds of the variables. For a simplicial set, the faces are defined by the so-called face graph. Let $I := \{1, 2, \ldots, |W|\}$ be the index set of vertex set W, then any subset $J \subset I$ corresponds to a face of a simplicial set $\mathrm{conv}(W)$.

For a non-simplex polytope, this is not the case. It is possible to verify a subset $W_J := \{w_j \mid j \in J\}$ of W is a face φ by the following result. Let c be the centroid of W_J. Consider the following LP problem:

$$\min \sum_{j\in J} \lambda_j \quad \text{s.t.} \quad \sum_{j\in I} \lambda_j w_j = c, \quad \sum_{j\in I} \lambda_j = 1, \quad \lambda_j \geqslant 0,\ j \in I. \tag{4}$$

Proposition 1. *Consider polytope* $q = \text{conv}(W)$ *and a polytope* $\varphi = \text{conv}(W_J)$ *of the subset* W_J *of vertices* W *of* q. *If the optimal objective function value of* (4) *is* 1, *then* φ *is a face of* q.

Proof. If the objective function outcome of (4) is 1, then there does not exist another convex combination including a vertex $w \notin W_J$ and $\varphi = \text{conv}(W_J)$ is a face of q. □

Using Definition 2 directly would imply using (4) and storing all faces φ of q with their dimension. Checking a face $f \subset \varphi$ requires solving (4) taking the centroid c of f and the subset J which defines φ. Checking this for all faces φ does not sound practical. There are some simple rules that can be derived from properties.

Proposition 2. *Given the feasible area* q, *an* m*-polytope subset* p *and its two border* $(m-1)$*-facets* f_j *and* f_i *with a nonempty intersection. The intersection face* $f_j \cap f_i$ *is a border* $(m-2)$*-facet of* f_j *and* f_i.

Proof. Both facets f_i and f_j are border, thus they are enclosed by $(m-1)$-facets of q, φ_j and φ_i, respectively. Therefore, $f_j \cap f_i$ is a subset of the $(m-2)$-facet $\varphi_j \cap \varphi_i$ of q, which is non-empty as $f_j \cap f_i \subseteq \varphi_j \cap \varphi_i$. □

As discussed in Hendrix *et al.* (2024), there are several ways to refine the partition, i.e. to split the subsets further. In this paper, we will focus on bisecting the subset along the widest coordinate of its interval hull, following traditional methods in iBB. To avoid dealing with the face set, we will keep track of border levels with a focus on facets and edges of a subset. Notice that all faces of the feasible set q are border. The first polytope subset is $p = q$. Its facets f have border level $\dim(f) = \dim(q) - 1$. In our procedure, we will keep track of the border status of facets and edges of polytope subsets p in the division (see Section 4.2), and update them when p is reduced, see Section 4.6.

3.2. *Mathematical Properties on Monotonicity Rejection or Reduction of a Polytope*

Consider two points $x, y \in p$, $x \neq y$ that define a direction from x: $d = y - x$, over polytope p.

Definition 4. If $\forall x \in p$, $\frac{\partial}{\partial \mu} h(x+\mu d) > 0$ for $\mu = 0$, direction d is called monotonically increasing on p. Monotonically decreasing, non-decreasing and non-increasing directions are defined similarly.

As we consider continuously differentiable functions, monotonically increasing follows from $d^T \nabla h(x) > 0$, Hendrix and G.-Tóth (2010). Let $\nabla h(p)$ be the range of $\nabla h(x)$,

$\forall x \in p$. Then $d^T \nabla h(p) := [\underline{d^T \nabla h(p)}, \overline{d^T \nabla h(p)}]$ is an interval with bounds containing all directional derivative values $d^T \nabla h(x)$, $x \in p$ where d is a direction in p. In this study, we use Interval Automatic Differentiation over $\square p$, to get bounds of the derivatives of the objective function over p, $\nabla h(p) \subseteq \boldsymbol{g}(\square p) := [\underline{\boldsymbol{g}}, \overline{\boldsymbol{g}}]$, see Moore *et al.* (2009), Rall (1981). Then, the corresponding directional derivative $d^T \nabla h(p)$ is in the range of the interval inner product

$$d^T \nabla h(p) \subseteq d^T \boldsymbol{g} = \left[\underline{d^T \boldsymbol{g}}, \overline{d^T \boldsymbol{g}}\right] = \left[\sum_{i=1}^{n} \min\{d_i \underline{\boldsymbol{g}}_i, d_i \overline{\boldsymbol{g}}_i\}, \sum_{i=1}^{n} \max\{d_i \underline{\boldsymbol{g}}_i, d_i \overline{\boldsymbol{g}}_i\}\right]. \tag{5}$$

Notice that overestimation (a less tight enclosure) of p by $\square p$ is larger as p deviates more from $\square p$. This increases the overestimation of $d^T \nabla h(p)$ in $d^T \boldsymbol{g}$. Let $\boldsymbol{dd} := d^T \boldsymbol{g}$ denote the interval directional derivative enclosure. If $0 \notin \boldsymbol{dd}$, then d is a monotonic direction on p.

In Casado *et al.* (2025), we focus on the question how to find monotonic directions in sBB. The most important impact is that, as for simplices (Casado *et al.*, 2022), the relative interior of the full or non-full dimensional polytopal partition set cannot contain an optimum.

Proposition 3. *Let $p \subseteq q$ be a partition set. If $\exists x, y \in p$, such that direction $d = y - x$ has corresponding directional derivative bounds $0 \notin [\underline{d^T \nabla h(p)}, \overline{d^T \nabla h(p)}]$ then* $\mathrm{rint}(p)$ *does not contain a global minimum point of* (1).

Proof. Consider $z \in \mathrm{rint}(p)$. As z is in the relative interior, there exists a feasible direction d in which lower function values can be found, i.e. $\exists \varepsilon \in \mathbb{R}^+$ small enough, such that $z + \varepsilon d \in p$ and $h(z + \varepsilon d) < h(z)$. So z cannot be a minimum point of h. □

Corollary 1. *Let $p \subseteq q$ be a partition set in a branch and bound algorithm. If the conditions of Proposition 3 apply and $p \subset \mathrm{rint}(q)$, then p can be rejected.*

The main consequence of Proposition 3 is that a non-interior subset p can be replaced by the collection of its facets, which basically have a lower polytope dimension and fewer vertices. This is mainly a theoretical observation. Keeping track of the facets of a polytope is complex. We will come back to this issue. One typical observation following Corollary 1 is the following.

Corollary 2. *Let f be a facet of $p \subseteq q$ in a branch and bound algorithm. If the conditions of Proposition 3 apply to f and $f \subseteq \mathrm{rint}(q)$, then f can be rejected.*

We now focus on the direction from a vertex of p pointing towards the relative interior, $d = y - v$, $v \in V$.

Proposition 4. *Let $p \subseteq q$ be partition set and v a vertex of p. If $\exists\, y \in \text{rint}(p)$, such that $(y - v)^T \nabla h(p) < 0$, then for all faces f of p which include vertex v, $\text{rint}(f)$ cannot contain a global optimum point.*

Proof. Consider $d = y - v$ and let $z \in \text{rint}(f)$. The function value $h(z + \varepsilon d)$ is lower than $h(z)$ for $\varepsilon > 0$ as long as $z + \varepsilon d \in p$, because d is a decent direction. This means, we have to show that $d = y - v$ is a feasible direction. For this, we should find a value for $\varepsilon \in \mathbb{R}^+$ small enough to show $z + \varepsilon(y - v) \in p$. Without loss of generality, number the vertices of p as $v_1, \ldots v_{|V|}$ with $v = v_1$. Notice that f is a convex hull of a subset of those vertices. According to the definition of relative interior, we have that $y = \sum_{j=1}^{|V|} \mu_j v_j$ with $\mu_j > 0$. Let $z = \sum_{j=1}^{|V|} \lambda_j v_j$ with $\lambda_1 > 0$ be a point in the interior of a face which contains v_1. This means that for all $\varepsilon \in \left(0, \frac{\lambda_1}{1-\mu_1}\right]$, point x from z in direction d, i.e. $x = z + \varepsilon d = (\lambda_1 + \varepsilon(\mu_1 - 1))v_1 + \sum_{j=2}^{|V|}(\lambda_j + \mu_j \varepsilon)v_j$ is feasible and has a function value $h(x)$ lower than in point z. Hence, $z \in \text{rint}(f)$ cannot be a minimum point of f. □

Corollary 3. *Let v be a vertex of $p \subseteq q$ in a pBB algorithm. Consider the conditions of Proposition 4 fulfilled for vertex v of p. We remove $\text{rint}(p)$ by replacing p by the set of facets of p following Proposition 3. Considering a vertex v implies that facets f with $v \in f$ can be left out of consideration.*

The argument is that the relative boundary of f consisting of faces not containing v may contain a global minimum point. However, the global minimum point is enclosed in the relative boundary of other facets that do not include v. One should take care in not applying this concept for several vertices simultaneously in all cases.

Corollary 4. *Corollary 3 can be applied considering more than one vertex satisfying the conditions for 2-polytopes. For larger dimensional polytopes, considering more than one vertex may lead to removing the minimum if the lower dimensional border faces are not taken into account.*

Figure 2 illustrates the consequences of Corollary 4.

There are several ways to find monotonic directions. In Hendrix *et al.* (2024), we extended the earlier analysis of Casado *et al.* (2021) for simplicial partition sets to polytopal partition sets. An easy way to start is to focus on the directions $d = c - v$ from a vertex to the centroid and the directions between vertices. In Hendrix *et al.* (2024), we show that the best direction in sense of highest or lowest directional derivative bound can be found solving a linear program. This may not be computationally efficient as discussed in Casado *et al.* (2025).

4. Main Rules of the Polytopal Branch and Bound Algorithm

Following the general schema of Algorithm 1, we first sketch the difficulty of the polytopal division rule by bisecting the widest coordinate of $\square p$.

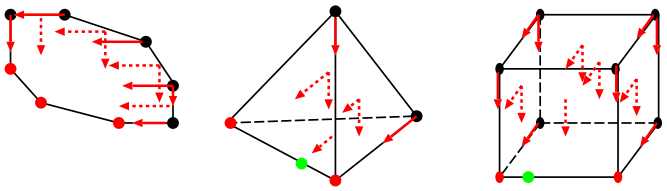

Fig. 2. Red arrows correspond to a negative directional derivative from a black vertex; dashed versions start at the centre of facets that can be removed. Faces with red vertices are those to reduce to. The green dot corresponds to a global minimum point. For the 2-polytope (left graph), we can remove all facets (edges) with a black dot, i.e. remove facets having a vertex with negative $\boldsymbol{dd}$ from it. For the 3-polytopes, we can only use one black vertex to discard facets with it, otherwise we might remove the global minimum. In sBB we can remove all black vertices with negative $\boldsymbol{dd}$, because remaining vertices form a face (middle graph). This is not always the case for polytopes. For instance, removing one vertex of a cube, the remaining vertices do not form a convex closed polytope.

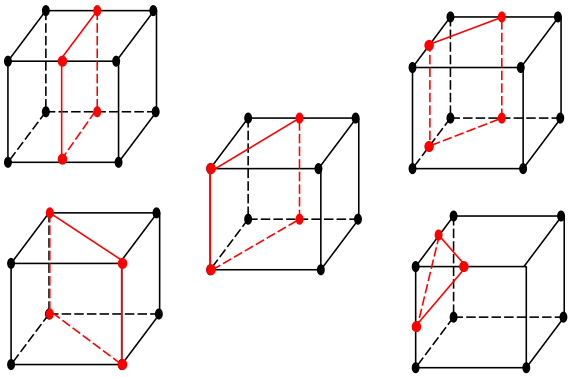

Fig. 3. Division of a cube. Notice that we can rotate the cube to have a component-wise division. Cutting facet f^c in red is a 2-facet. Having three vertices at c implies three edges of f^c. Having four vertices at c, some combinations of two of them do not correspond to an edge of f^c, i.e. an edge in the convex hull of vertices.

4.1. *Division Rule: Coordinate-Wise Bisection*

In this study, we describe a polytope $p = \{V, E, F, m\}$ as a set of vertices, edges, facets and its dimension, respectively. By performing a bisection, we generate two polytopes, one at left $p^\ell = \{V^\ell, E^\ell, F^\ell, m\}$, and one at right $p^r = \{V^r, E^r, F^r, m\}$ of the midpoint of the widest coordinate of $\square p$. One of the tasks to perform is to determine the cutting facet f^c. Determining the vertices of f^c is not difficult, but to determine its edges can be a challenge, because a pair of vertices in f^c might not be connected by an edge. We first show the difficult case and later on the easier ones in determining f^c.

As illustration, Fig. 3 shows some cases of the cutting facet f^c in red for a cube bisection. The cubes may be rotated to illustrate a component-wise division where internal division is not in the middle of the widest component. Notice that in general we divide edges, but there are cases where we do not, see left-down graph of Fig. 3.

Table 2
Definitions and initial sets for a bisection.

$p = \{V, E, F, m\}$	polytopal subset defined by sets of vertices V, edges E, facets F and dimension m.
$e = (u, v), u, v \in V, e \in E$	edge of p.
$f_j = \{V_j, E_j\}, f_j \in F$	$(m-1)$-facet j of p determined by edges $e \in E_j \subset E$.
$c = \text{mid}(\square p_i)$	middle point of $\square p_i$, i is the widest coordinate of $\square p$.
$V^c = \{v \in V : v_i = c\}$	set of vertices at c.
$V^\ell = \{v \in V : v_i < c\}$	set of vertices of p at left of c.
$V^r = \{v \in V : v_i > c\}$	set of vertices of p at right of c.
$E^c = \{e = (u, v) \in E : u, v \in V^c\}$	edges at c.
$E^\ell = \{e \in E : u \in V^\ell, v \in V^\ell \cup V^c\}$	edges at left of c.
$E^r = \{e \in E : u \in V^r \cup V^c, v \in V^r\}$	edges at right of c.
$E^d = \{e \in E : u \in V^\ell, v \in V^r\}$	edges to be divided.
$F^\ell = \{f \in F : \forall e \in f,\ e \in \{E^c \cup E^\ell\}\}$	set of facets at left of c, do not have to be divided.
$F^r = \{f \in F : \forall e \in f,\ e \in \{E^c \cup E^r\}\}$	set of facets at right of c, do not have to be divided.
$F^d = \{f \in F : \exists\, e \in f,\ e \in E^d$ **or** $\exists e_i, e_j \in f,\ e_i \in E^\ell,\ e_j \in E^r\}$.	set of facets to be divided.

Table 2 shows the notation for the used elements and the sets to be determined as starting step before performing a bisection. Facets of a facet f_j of a polytope p have to be determined only if f_j is treated as a polytope, which will occur only after a reduction to it, see Section 4.5. Therefore, we do not need to identify the facets of a facet in the division of p, but only the set of vertices and edges a facet has.

The difficulty is illustrated by the labelling provided in Table 2. After the identification of the sets, the aim is to divide edges at E^d generating new vertices at c that are added to V^c. For each divided edge, we generate a new vertex at c and two new edges that are stored at new sets:

- $E^{d\ell}$: set of edges at left of c, as left result of division of edges in E^d.
- E^{dr}: set of edges at right of c, as right result of a division of edges in E^d.

An edge $e \in E^d$ belongs to one or more facets in F^d. We generate two new sets:

- $F^{d\ell}$: contains facets generated from a facet f in F^d by copying the edges on the left of c and the left part of divided edges of f.
- F^{dr}: similarly taking the edges at right of c and the right part of divided edges.

In the case where the facet to be divided has no edges to divide (see top and bottom facets of the cube at left-down graph of Fig. 3), we have to separate edges generating two facets; one with edges at left and store them in $F^{d\ell}$ and one with edges at right and store them in F^{dr}. At this point, we have open facets sets at the left $F^{d\ell}$ and right F^{dr}. Figure 4 shows an example of sets $F^{d\ell}$ and F^{dr} of a cube bisection.

What remains to identify are edges E^c of f^c after construction of the complete vertex set V^c. The difficult part of a bisection algorithm based on only vertices is that it requires checking, whether a pair of vertices (u, v) is an edge of a polytope, in our case for f^c. In López and Duval (2012), one can find algorithms to compute edges from a vertex set. They are based on finding supporting hyperplanes using linear programming (LP). A simple procedure is to check by LP (4) whether the midpoint $c = \frac{v_\ell + v_k}{2}$ of two vertices v_ℓ and

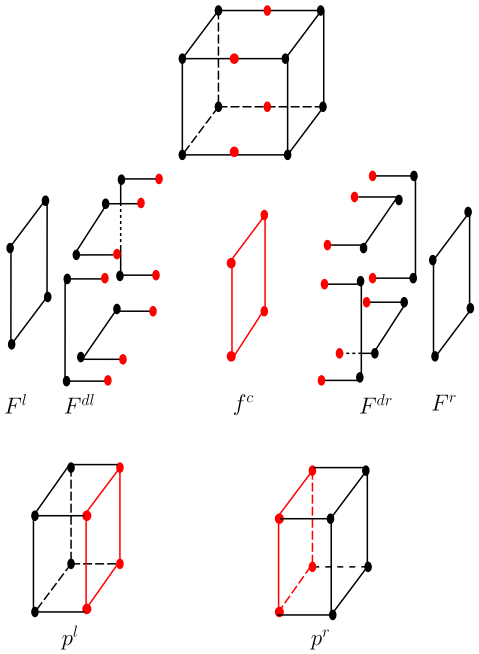

Fig. 4. Example of a cube division to show the facet sets during the process.

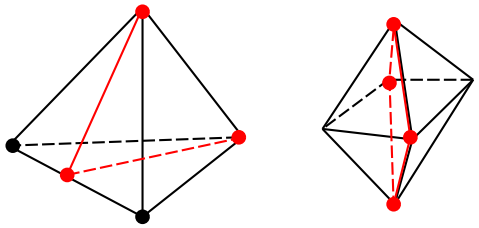

Fig. 5. Division of 3-polytopes. Notice that we can rotate the polytope to have a component-wise division. Cutting facet f^c in red is a 2-facet. The left graph shows a Longest Edge Division which divides an edge. Having three vertices at c all are connected in f^c. There are neither divided facets, nor edges of p in the bipyramid example at the right. So, $F^{d\ell} = F^{dr} = \{\emptyset\}$.

v_k is on a facet of p. Such procedures require running an LP routine. This does not seem attractive in an iterative process for each pair of vertices of f^c in a spatial B&B search. Moreover, in our experience, LP routines are getting sensitive to numerical errors when the subsets are getting small.

We use a different approach in Algorithm 2 to determine the edges of k-facet f^c, for the most complicated cases, $|V^c| > 3$, $k > 1$. Easier cases will be described later in this section, see graphs of Fig. 5. Algorithm 2 uses a set F^{dr} to determine all edges $e \in E^c$. This could be $F^{d\ell}$ as well, because both need the same edges of facet f^c. It starts to determine the set of vertices at c for each facet $f_j \in F^{dr}$. We also need to keep track of the

Algorithm 2 Update E^c, $f \in \{F^{dr} \cup F^{d\ell}\}$, F^ℓ, F^r and generate $f^c = \{V^c, E^c\}$

Require: $|V^c| > 3$, E^c, F^{dr}, $F^{d\ell} \neq \{\emptyset\}$, F^ℓ and F^r

1: $R = \{\emptyset\}$ ▷ A set of (U, I) pairs with central vertices and their facet indices
2: **for** each $f_j \in F^{dr}$ **do** ▷ Facet wise investigation of central vertices
3: $U_j = V_j \cap V^c$ ▷ Vertices of a facet j at cutting facet
4: $I_j = \{j\}$ ▷ Index of the facet U_j vertices are in
5: Add $r = (U_j, I_j)$ to R
6: **while** $\max_{(U,I)\in R} |U| > 2$ **do**
7: $T = \{\emptyset\}$ ▷ Temporal set as R, storing $\cap, \cup$ of $r \in R$ components
8: **for all** $r_i \neq r_j \in R$ **do** ▷ Let $r_i = (U_i, I_i)$, $r_j = (U_j, I_j)$
9: $U = U_i \cap U_j$ ▷ Intersection of vertices among faces
10: $I = I_i \cup I_j$ ▷ Indices of facets U vertices are in
11: **if** $|U| < 2$ **then continue** ▷ They do not share an edge on c.
12: **if** $\exists\, t = (U_r, I_r) \in T : U = U_t$ **then** ▷ Remove U duplicates in T
13: $I_t = I_t \cup I$ ▷ U might exists in different facets
14: **else**
15: Add (U, I) to T
16: $R = T$
17: **for** $r \in R$ **do** ▷ R has all edges at c and the indices of divided facets they are
18: $e = (u, v),\ u, v \in U$ ▷ $r = (U, I)$
19: **if** $e \notin E^c$ **then** ▷ Divided facets have already edges at c.
20: Add e to E^c
21: **for** each index $i \in I$ **do**
22: Add e to E_i^{dr} ▷ $f_i^{dr} = \{V_i^{dr}, E_i^{dr}\} \in F^{dr}$
23: Add e to $E_i^{d\ell}$ ▷ $f_i^{d\ell} = \{V_i^{d\ell}, E_i^{d\ell}\} \in F^{d\ell}$
24: Add F^{dr} to F^r
25: Add $F^{d\ell}$ to F^ℓ
26: Add $f^c = \{V^c, E^c\}$ to F^ℓ and F^r

indices of the facets the vertices are in. Thus, we use a set R with elements $r = (U, I) \in R$ having two components: U, the set of vertices at c and I, the set of indices of the facets the vertices at U belong to. In line 2 of Algorithm 2, we store the vertices at c and the index of the facet for each $f_j^{dr} \in F^{dr}$ in R. The main computational burden of the procedure is in the while in line 6 of Algorithm 2. This computational burden increases with the dimension of f^c. It is based on the iterative intersection of vertices U and the union of the corresponding indices of the facets I for elements in R, while at least two vertices are in U for each $r \in R$. Intersections with less than two vertices are not considered (they can not generate an edge) and we also have to take care to avoid duplicates in the process. This process can be seen as the traversal of the face-graph of f^c, having f^c at the top level, and its facets (determined by their vertices) in the next lower level. The process moves down, level by level, until the level with edges is reached. Notice that the while loop will not be

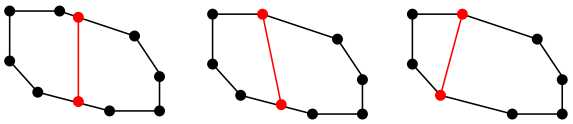

Fig. 6. Division of a 2-Polytope. Notice that we can rotate the polytope to have a component-wise division. The two vertices at c are new on the left, existing ones on the right, and mixed in the middle. They define the cutting interior 1-facet f^c in red.

executed in the illustrative example in Fig. 4, because each U_j has exactly two elements, so we find all edges right away. Finally, Algorithm 2 adds new edges to the corresponding open facets at $F^{d\ell}$ and F^{dr} to close them. Then, $F^{d\ell}$ is added to $F^\ell \subset p^\ell$ and F^{dr} to $F^r \subset p^r$. Facet f^c completes both F^ℓ and F^r. The bottom graphs of Fig. 4 show an example of resulting sets p^ℓ and p^r.

As mentioned above, there exist easier cases to determine cutting facet f^c. If the set of facets to be divided F^d is empty (see right graph of Fig. 5), vertices in V^c and edges in E^c of f^c are identified. In this case, the set of facets in p^ℓ is $F^\ell \cup f^c$ and the set of facets in p^r is $F^r \cup f^c$. When the number of vertices in V^c after dividing edges is three, they must be all connected in f^c, see left graph of Fig. 5. So, Algorithm 2 is not needed.

For 2-polytopes, it is easy to determine cutting facet f^c, because f^c is a segment connecting two vertices, which can be new or not. Figure 6 shows some examples.

Finally, if p is 1-polytope (segment) the cutting 0-facet is the new vertex.

4.2. *Border Level of Edges and Facets in the Division*

For keeping track of the border level, we can exploit the following rules:

- Facets f and edges e of feasible set q, an m-polytope, are border and have border-level $\boldsymbol{bl}(f) = m - 1$ and $\boldsymbol{bl}(e) = 1$.
- Each edge and facet going into p^ℓ or p^r (or both) unchanged keeps its border level.
- Edges $e^{d\ell}$ and e^{dr} generated by bisecting edge e^d inherit the border level of the divided edge, $\boldsymbol{bl}(e^{d\ell}) = \boldsymbol{bl}(e^{dr}) = \boldsymbol{bl}(e^d)$.
- Generated new facets $f^{d\ell}$ and f^{dr} inherit the border level of divided facet f^d.
- Facet f^c used to cut subset p is in the interior of p; its border level is $\boldsymbol{bl}(f^c) = \boldsymbol{bl}(p)$.
- A new edge e^c in the cutting k-facet f^c has the same border level as that of the split facet $f^d \in p$, for $k \geqslant 2$. For $k = 1$, $\boldsymbol{bl}(f^c) = 2$ because it is an edge in the interior of p. Notice that f^d can be a cutting facet in earlier iterations or a division of it, compared to current node p.

4.3. *Bounding Rule*

Interval Arithmetic and Automatic Differentiation are easy choices to get bounds of the objective and its derivatives over a box. In a very summarized way, Interval Arithmetic replaces reals by intervals and real operations by interval ones. The very first objective of Interval Arithmetic was to avoid rounding errors of Computational Arithmetic. For

polytopes (and simplices) we can use the subset interval hull to get bounds of the objective function over p: $h(p) \subseteq \boldsymbol{h}(\square p)$ and its derivatives: $\nabla h(p) \subseteq \boldsymbol{g}(\square p)$. So, iBB tools can be used in pBB.

The so-called centred form of the Taylor extension may improve the bounds of the direct extension from real arithmetic to intervals $\boldsymbol{h}(\square p)$ by using first-order derivatives $\boldsymbol{h}_c(\square p) = h(c) + \boldsymbol{g}(\square p)(\square p - c)$, for some $c \in \square p$, see G.-Tóth *et al.* (2021) for more details on bounding rules. We will use the evaluation of the centre of p in $\boldsymbol{h}_c(\square p)$, instead of the centre of $\square p$, which usually obtains a smaller maximum distance from centre to vertices, which is required in the $\boldsymbol{h}_c(\square p)$ computation.

4.4. *Selection and Termination Rules*

As a **selection rule**, we choose the polytope subset $p \in \Lambda$ with the lowest updated lower bound $\underline{\boldsymbol{h}}(\square p) = \max\{\underline{\boldsymbol{h}}(\square p), \underline{\boldsymbol{h}_c}(\square p)\}$ to be processed in the next iteration of Algorithm 1.

Using the centred form, the centre of p is evaluated. It can update the best upper bound of the solution so far, the so-called incumbent represented by $\overline{\boldsymbol{h}}$. Symbol $\underline{\boldsymbol{h}}$ denotes the lower bound of the solution, which is $\underline{\boldsymbol{h}}(\square p)$ for the selected polytope.

Algorithm 1 **terminates** when $\overline{\boldsymbol{h}} - \underline{\boldsymbol{h}} \leqslant \alpha$. The interval $[\underline{\boldsymbol{h}}, \overline{\boldsymbol{h}}]$ and the remaining inclusion set Λ, which contains the minima, are the output of Algorithm 1.

4.5. *Rejection and Reduction Rules*

The theoretical results give rise to concrete tests that can be used to reject subsets from further consideration or to reduce them to lower dimensional subsets. First of all, the **RangeUp test** rejects a subset p if $\underline{\boldsymbol{h}}(\square p) > \overline{\boldsymbol{h}}$, see Section 4.4. If the test fails, we can evaluate the centred form to test $\underline{\boldsymbol{h}_c}(\square p) > \overline{\boldsymbol{h}}$. The application of the RangeUp test to already evaluated subsets at Λ is the so-called **CutOff test**.

We now focus on exploiting the theoretical monotonicity properties discussed in Section 3.2 to decide on rejection and reduction. A necessary condition to reject or reduce a polytope p based on monotonicity is $0 \notin \boldsymbol{g}(\square p)$. It is important to differentiate if the m-polytope p is full dimensional ($m = n$) or not ($m < n$), and if it has border facets or not. For m-polytopes, $m < n$, condition $0 \notin \boldsymbol{g}(\square p)$ is not sufficient and we need to find a directional derivative $0 \notin \boldsymbol{dd} = d^T \boldsymbol{g}(\square p)$ in p, see (5). We will use the notation $\boldsymbol{dd}^+$ for $\underline{\boldsymbol{dd}} > 0$ as positive directional derivative and $\boldsymbol{dd}^-$ for $\overline{\boldsymbol{dd}} < 0$. In this study, we limit the search to directions from vertices of p, $\boldsymbol{dd}_v$, $v \in p$ to the centre and to other vertices of p. Figure 7 shows a hypothetical example where both $\boldsymbol{dd}_v^-$ and $\boldsymbol{dd}_v^+$ exist in v. We are interested mainly in $\boldsymbol{dd}_v^-$, see Corollary 4. Hence, negative directional derivative $\boldsymbol{dd}_v^-$

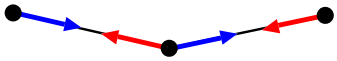

Fig. 7. Blue arrows correspond to positive directional derivative $\boldsymbol{dd}_v^+$ and red to negative $\boldsymbol{dd}_v^-$. The example can be a partial view of vertices and edges of a larger polytope.

could replace the information about monotonic directional derivative in v when it is $\boldsymbol{dd}_v^+$. Notice that a positive directional derivative $\boldsymbol{dd}_v^+$ from v to vertex u implies having a negative $\boldsymbol{dd}_u^-$.

The rejection and reduction cases when $0 \notin \boldsymbol{g}(\Box p)$ are as follows.

1. Polytope subset p has dimension $m = n$ (full dimensional):
   a) If p does not have border facets, it is rejected, see Corollary 1.
   b) If p has border facets, it is reduced to its border facets, see Corollary 2. We avoid the search in some or all border facets by finding a negative $\boldsymbol{dd}_v^-$, as is done for non full-dimensional p cases 2(b)iiA and 2(b)iiB below.
2. Polytope subset p has dimension $m < n$ (non full dimensional):
   a) If no vertex v is found with $0 \notin \boldsymbol{dd}_v$, p is stored in Λ for further processing.
   b) If a vertex v is found with $0 \notin \boldsymbol{dd}_v$:
      i. If subset p does not have border facets, it is rejected, see Corollary 1.
      ii. Subset p with border facets is reduced to its border facets, see Corollary 2. The search of some or all border facets can be avoided, if a negative direction $\boldsymbol{dd}_v^-$ is found.
         A. A negative direction $\boldsymbol{dd}_v^-$ was found and $m = 2$. We ignore border facets with vertex v. More than one of such a vertex v can be used, see Corollary 4 and left graph of Fig 2.
         B. A negative direction $\boldsymbol{dd}_v^-$ was found and $m > 2$. We ignore border facets with vertex v. Only one of such a vertex v can be used, see Corollary 4.

In case 2(b)iiB, we can stop after one negative direction $\boldsymbol{dd}_v^-$ is found or look for all of them and apply Corollary 4 using the vertex that removes more border facets. If all border facets are removed, p is not reduced but rejected. If p is a 1-polytope (segment), it can only be reduced to a vertex of the initial feasible set q, which is a border 0-facet of p.

Consider reduction of subset m-polytope $p = \{V, E, F, m\}$ to a border facet $f_j \in F$. Then, f_j is evaluated, see Section 4.3. Before storing f_j in Λ, the RangeUp test is checked. If it passes the test, we have to store f_j as a polytope, $f_j = \{V_j, E_j, F_j, m-1\}$. This requires to determine the set of $(m-2)$-facets $f_i^{f_j} \in F_j$ of f_j. Each $f_i^{f_j}$ is obtained by a nonempty intersection of polytopes $f_j \cap f_i$, $j \neq i$, $f_j, f_i \in F \in p$.

We observed in the experiments that when selecting the next reduced-evaluated-RangeUp p from Λ, it is worth to try to reduce p again with the new derivative bounds, instead of directly dividing it. We do not apply more than one reduction-evaluation-RangeUp before to store p in Λ, because this would affect the selection rule (see Section 4.4), which might increase the computational burden of the algorithm. However, by storing p after a reduction-evaluation-RangeUp, the CutOff could remove p from Λ in a next iteration.

4.6. *Updating the Border Level and Setting the Border Status After Reduction*

The determination of the border level of facets and edges after a polytope division was discussed in Section 4.2. Lemma 1 determines the border status (border level) of a facet. Facets generated by successive division of a facet f maintain the border status of f.

All facets of feasible set q are border. A first non-border cutting facet f^c is generated by the bisection of the widest component of $\square q$. The algorithm will never reduce to a cutting facet, or facets generated by a successive division of a cutting facet, because they are not border and are labelled as non-border after a division.

Consider an m-polytope $p = \{V, E, F, m\}$ to be reduced to a border $(m-1)$-facet $f_j = \{V_j, E_j, F_j, m-1\}$. As Section 4.5 discussed, each $(m-2)$-facet $f_i^{f_j}$ of f_j is obtained by the non empty intersection $f_j \cap f_i$, $j \neq i$, $f_j, f_i \subset F \subset p$. To determine the border status of the facets $f_i^{f_j}$ of f_j, Proposition 2 states that if both f_i and f_j are border, then $f_i^{f_j}$ is border.

The next question concerns the border status of the facets $f_i^{f_j} = f_j \cap f_i$ after reduction of p to f_j and f_i is not border due to being a cutting facet or a division of it. The problem is that face $f_i^{f_j} \subset p$ is a facet of both f_j and f_i, and it can be border. Examples are provided at the bottom left graph of Fig. 3 and right graph of Fig. 5, where there exist border edges in f^c. In general, f_i can have a border face. So we need some tools to determine its border status. Proposition 5 characterizes a non-border facet without border faces.

Proposition 5. *Given m-polytope $p = \{V, E, F, m\}$ and a cutting $(m-1)$-facet f^c. If all vertices v of f^c are new ($v \notin V$), f^c has no border faces.*

Proof. If all vertices of f^c are new, all its edges are new ($e \notin E$) as well. Thus, all faces built from the new edges are new. New faces with all new sub-faces are in the relative interior of p, i.e. every k-face of f^c is in the relative interior of a $(k+1)$-face of p. So, they constitute f^c by construction. □

Proposition 5 also shows that facet f^c under this conditions does not contain a face of p. If $E^c = \emptyset$ in the input of Algorithm 2 and f^c has no vertex of q, then f^c has no border faces. Therefore, $f_i^{f_j}$ and any of its subsets generated by division are not border, as well.

Corollary 5. *Given face $f_i^{f_j} = f_j \cap f_i$ after a reduction of p to a border facet f_j. If non-border facet f_i of p has no border faces, then face $f_i^{f_j}$ is not border.*

However, if f^c contains faces of p, the problem is to identify them and to determine they are not border and if they are also in another border facet. In our algorithm, we try to simplify this task using the condition of Proposition 6.

Proposition 6. *Given an m-polytope p with $m \geqslant 2$ and $(m-1)$-facet f of p, which has not yet been labelled as border or not. If f is border, then it must have a vertex $v \in f$ in at least k edges $e \in f$ with $\boldsymbol{bl}(e) \leqslant (m-1)$.*

Proof. Each vertex v of an m-facet is at least in m edges in order to have a facet with dimension k. For $m = 1$, if the $\boldsymbol{bl}(e) = 1$, e is one edge of q or was generated by its division, so it is border. For $m = 2$, we need at least two edges e_1 and e_2 sharing a vertex

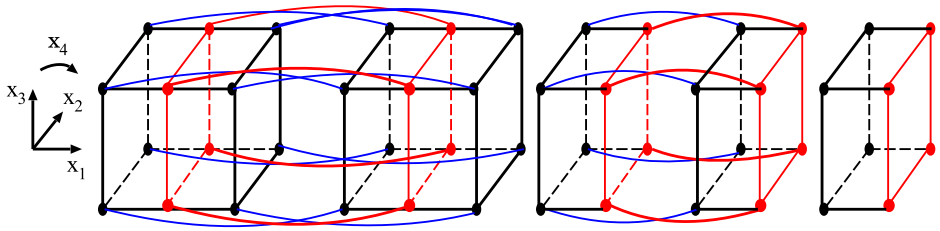

Fig. 8. Consider feasible set q to be a 4-cube polytope at the left graph. This means its facets are 3-cubes. We divide q by the middle of coordinate x_1. The interior cutting 3-cube facet f^c in red has border level 4 like q because f^c is in the relative interior of q. So, it is not border in the generated q^ℓ and q^r polytopes. The border level of an edge of f^c is the smallest border level of the facets it belongs to. In this case, they all have a level of 3. Let 4-polytope q^ℓ in the middle graph be reduced to its border 3-facets with border level 3, due to a negative directional derivative $\boldsymbol{dd}_v^-$ in direction $-x_4$. Consider one of them to be the 3-polytope p at the right graph. Edges that were generated at 3-facets of q are now in 2-facets of p. So, we have to decrease their border level by one.

with $\boldsymbol{bl}(e_i) \leqslant 2$ in order to have an area in a 2-face of q. This can be extended to any k in the same way. □

Proposition 6 is only relevant for a face $f_i^{f_j}$ after a reduction, because all facets were already labelled in a division. It provides a necessary, but not sufficient test for facet $f_i^{f_j}$ to be border. If the condition is not fulfilled, the facet is not border.

Moreover, Proposition 2 says that $f_i^{f_j}$ is border if both f_i and f_j are border. In the algorithm, we test the condition of Proposition 6 on $f_i^{f_j}$ only when $f_i^{f_j}$ was not labelled as border by Proposition 2 and it could not be labelled as non-border by Corollary 5.

Proposition 6 requires to keep track of the edge border levels after reducing to a border facet f_j. Rules in Section 4.2 show that the border level of an edge can be inherited from a divided edge or from the facet it is generated in. By reducing an m-polytope p to a border $(m-1)$-facet f_j, we diminish the dimension by one. This implies that, when $m > 3$, we also have to decrease the border level of edges of p with $\boldsymbol{bl}(e) \geqslant (m-1)$, before they are inherited by the border facet f_j. Figure 8 shows an illustrative example.

Figure 8 provides an example of Proposition 5 in the red cube f^c. Figure 8 also aids to observe that Proposition 6 provides a necessary, but not sufficient condition to label a facet $f_i^{f_j}$ of the border facet f_j to which we reduce. All 2-faces of f^c are not border. They are in the relative interior of the corresponding 3-facet of q. Consider the 4-cube q^ℓ in the middle graph. It has eight 3-cube facets. One of them is the cutting facet of q in red. If we reduce q^ℓ to the 3-cube facet f_j at the left, we have to determine its facets $f_i^{f_j}$ by intersecting f_j with the other 3-cube facets of q^ℓ. All facets of q^ℓ are border apart from the red 3-cube facet f_i. So, all but one $f_i^{f_j}$ are border and we have to determine the border status of $f_i^{f_j} = f_j \cap f_i$, which is the red square in the right graph. Face $f_i^{f_j}$ is in the relative interior of q and satisfies the condition of Corollary 5. So, it is labelled as non-border in pBB because f^c has no border faces. Although, Proposition 6 will not be checked by pBB, it is satisfied, i.e. there exists one vertex in two edges with border level 2. So, Proposition 6 does not provide a test to guarantee the facet is border.

Table 3
Test problems, see Appendix A. The number of vertices v, edges e, and facets f are given in the description column for non-simplicial polytopes.

Instance	Description	n
2Pol	8v, 8e, 8f	3
3Pol	6v, 9e, 5f	3
4Pol	8v, 24e, 16f	6
Ex62	Karhbet example 6 over simplex 1	2
Ex62In	Karhbet example 6 over simplex 2	2

In case the condition of Proposition 6 is not satisfied, we label $f_i^{f_j}$ as border. The open question is whether a non-border $f_i^{f_j}$ exists not satisfying Corollary 5 and satisfying Proposition 6. In such case, we are labelling $f_i^{f_j}$ as border incorrectly and the algorithm may reduce to it. However, pBB still converges to the global minimum, because reduction is only to border facets and facets that might contain a global minimum are not discarded in the search. But pBB may evaluate polytopes that do not need to be tested.

5. Numerical Experiments

Algorithms were run on an Intel(R) Core(TM) i7-4770K CPU and 16GB of RAM running Fedora 42 Linux distribution. The algorithm was coded with g++ (gcc version 15.1.1) and uses http://verifiedby.me/kvKv-0.4.57 for Interval Arithmetic. Kv uses https://www.boost.org/boost libraries. Algorithms were compiled with `-O3 -DNDEBUG -DKV_FASTROUND` options.

We evaluated the algorithms on carefully designed experimental instances given in Table 3 and described in Appendix A. We are working on a larger benchmark, which does not exist yet for GO on polytope feasible sets to evaluate the dimension effect in the future in forthcoming papers.

5.1. *Experimental Settings*

Each algorithm has a different input for the same polytope q as feasible set:

iBBLC: The set of vertices W and the linear constraints defining q. The initial bounding box is the interval hull of W: $\square W$. Linear constraints representing variable bounds are always satisfied by $\square W$. So, they are not needed as input.

sBB: The simplicial partition of q. Each simplex is determined by its set of vertices.

pBB: Polytope $q = \{W, E, F, m\}$ defined by its set of vertices, edges, facets and its dimension.

In order to compare the three spatial B&B algorithms as fair as possible, the following B&B rules were taken for all algorithms.

Bounding: Each algorithm uses the interval centred form $\boldsymbol{h}_c$, using the centre of the set to get bounds of the real range of the objective function and its derivatives (see Sec-

tion 4.3). The centred form for sBB and pBB usually has a smaller maximum distance from the centre to any vertex.

Rejection/Reduction: All algorithms use the RangeUp, CutOff and Monotonicity tests. Due to monotonicity, the set may be reduced in dimension.

- iBBLC: reducing interval(s) $[\underline{x}_i, \overline{x}_i]$ to $\underline{x}_i$ or $\overline{x}_i$. Additionally, an infeasible box is discarded from the search.
- sBB: Uses directional derivatives; $\boldsymbol{dd}_v$ from vertex to centre and vertex to vertex. Additionally, a local search to find a $\boldsymbol{dd}_v^-$ is also used, see Casado *et al.* (2025).
- pBB: uses directional derivatives $\boldsymbol{dd}_v$ from vertex to centre and vertex to vertex, as outlined in Section 4.5.

Selection: Each algorithm selects the partition set with lowest lower bound of the objective function to be processed next, see Section 4.4.

Division:

- iBBLC: Bisects the first widest component of the box.
- sBB: Performs first longest edge bisection (LEB).
- pBB: Bisects the first widest component of the $\square p$, see Section 4.1.

Termination: All algorithms finish the search when $\overline{\boldsymbol{h}} - \underline{\boldsymbol{h}} \leqslant \alpha = 10^{-6}$, see Section 4.4.

The feasible area for all algorithms is the same. Some considerations about feasibility of a box in iBBLC are the following. A point is treated as a degenerated box in its feasibility evaluation. Having linear constraints $A\boldsymbol{x} \leqslant b$, for each slack $\boldsymbol{s}_j = A_j\boldsymbol{x} - b_j$, over a box $\boldsymbol{x}$, we have:

- $\overline{\boldsymbol{s}}_j \leqslant 0.0 : \boldsymbol{x}$ is feasible.
- $\underline{\boldsymbol{s}}_j > 0.0 : \boldsymbol{x}$ is infeasible.
- Otherwise : $\boldsymbol{x}$ is undetermined.

The linear constraints for the instances are provided in Appendix A. For the iBBLC algorithm we show if the minimum point x^* is feasible (F), infeasible (I) or undetermined (U). Additionally, we show the exponent of the distance of $\boldsymbol{s}_j$ to zero in scientific notation. Examples of this distance (which is different from interval distance to a point) are shown next. For instance, e-x in Ie-x is the exponent of the very first infeasible $\boldsymbol{s}_j$ given in $\underline{\boldsymbol{s}}_j$. In Fe-x, e-x is the minimum exponent for all feasible s_j, given in some $\overline{\boldsymbol{s}}_j$. If there exists one $\overline{\boldsymbol{s}}_j = 0.0$, we show F0 instead. In Ue-x, e-x is the minimum exponent for all limits $\underline{\boldsymbol{s}}_j$, $\overline{\boldsymbol{s}}_j$ of undetermined (might exist feasible $\boldsymbol{s}_j$) slacks $\boldsymbol{s}_j$.

Due to computational representation of numbers, two numbers are considered equal if their absolute difference is less than 10^{-12}. This is used to determine if a new vertex (point) already exists and has already been evaluated. This is also relevant for polytope division in order to determine if a cutting vertex already exits. Special care is required to the storage of a set S in Λ in an efficient way. We use an AVL tree, sorted by $\boldsymbol{h}(S)$. A linked list is used in an AVL tree node for equal values of different $\boldsymbol{h}(S)$. In the list, we use LIFO (last in, first out) to be more efficient in insertion and extraction. Interval $\boldsymbol{x} < \boldsymbol{y}$ when $\underline{\boldsymbol{x}} < \underline{\boldsymbol{y}}$ or ($\underline{\boldsymbol{x}} = \underline{\boldsymbol{y}}$ and $\overline{\boldsymbol{x}} < \overline{\boldsymbol{y}}$). Vertices are also stored in an AVL tree sorted by

Table 4
Numerical results. Column headers show the algorithms and the following notation. T: time (1e-2 precision in seconds), #S: number of generated subsets, #P: number of generated points. x^* is either F: Feasible, I: Infeasible, U: undetermined, as described in Section 5.1. MinAt shows in which face of q the minimum is. Green *: optimum at vertex reached by dimension reduction. Best results are in grey.

	iBBLC				sBB			pBB			
	T	#S	#P	x^*	T	#S	#P	T	#S	#P	MinAt
2Pol	0.01s	1 069	752	F0	0.01s	7	13*	0.01s	2	*10	v_5
3PolIn	0.01s	1 475	858	Fe0	0.03s	9 001	10 124	0.01s	1 415	3 122	Interior
3Pol$_{f_2}$	0.01s	611	1 015	Fe0	0.01s	553	667	0.01s	137	260	f_2 facet
3Pol$_{f_3}$	>5m			F0	0.01s	588	748	0.01s	438	709	f_3 facet
3Pol$_{e_8}$	0.60s	466 297	334 803	F0	0.01s	114	127	0.01s	54	92	e_8 edge
3Pol$_{v_6}$	0.01	559	399	F0	0.01s	44	*54	0.01s	13	*27	vertex v_6
4Pol	>5m			Ie-11	0.82s	58 790	59 195	0.01s	454	709	(v_3, v_4, v_5) face
Ex62	0.01s	2 687	1 916	F0	0.01s	67	101	0.01s	55	92	(v_1, v_2) edge
Ex62In	0.01s	163	150	Fe-1	0.01s	208	280	0.01s	103	196	Interior

their coordinates. Thus, no duplicates exist. We can remove a vertex when it is no longer in a set (to save memory) or let it stay forever (fewer vertex evaluations but more memory requirement). For this experimentation, we have chosen not to remove them.

5.2. *Numerical Results*

The considered instances are described in Appendix A. For sBB, non-simplicial polytopes are partitioned into simplicial subsets using the algorithm in Assarf *et al.* (2017). To use sBB for more than one simplex in the initial list of subsets, we had to extend it to deal with several simplices where not all facets are border initially. One should take care because, in such a case, sBB may discard a non-border facet which includes the solution, as the algorithm might find a negative $\boldsymbol{dd}_v^-$ to that non-border facet, see G.-Tóth *et al.* (2024).

Table 4 shows the results of the three algorithms for the designed set of instances. For non full dimensional instances 2Pol and 4Pol, see Appendix A.1 and A.3, no box generated by the iBBLC algorithm is feasible and the number of boxes increases significantly with the dimension. Although the execution time for all algorithms is less than 0.01s, iBBLC generates 150 times more subsets than sBB and pBB for the 2Pol instance. The differences are larger for the 4Pol instance: iBBLC could not solve it in less than 5 minutes and sBB generates thousands of simplices, but pBB only hundreds of polytopes.

Table 5 describes the meaning of the colours used in the illustrative Figures 9 to 13.

Figure 9 shows graphically the iBBLC execution over instance 2Pol. Yellow lines represent the linear constraints. There exist boxes rejected by infeasibility that are graphically overlapping the constraint, but they are actually not doing so. This visual effect is due to the graph only showing x_1, x_2 coordinates and there is more than one box sharing x_1 and x_2 components, see Fig. 14 in Appendix A.1. Even when no boxes in iBBLC are feasible, at least some evaluated centres of boxes, by the interval centred form, are feasible. Finding a feasible point is very important in constrained problems in order to use the RangeUp and CutOff tests. The refinement generates a lot of infeasible and undetermined boxes before reaching the termination criterion.

Table 5
Colours used in figures.

	Colour	Meaning
Elipsoids	Dark green	Set rejected by monotonicity test
	Light green	Set reduced to border facet(s)
	Brown	Set rejected by RangeUp test.
	Purple	Set rejected by CutOff test.
	Blue	Box rejected by infeasibility test.
iBBLC: box edge	Blue	$\boldsymbol{g}(\boldsymbol{x})$ is monotonically decreasing in x_i (increasing in $-x_i$).
	Red	$\boldsymbol{g}(\boldsymbol{x})$ is monotonically increasing in x_i (decreasing in $-x_i$).
	Black	$\boldsymbol{g}(\boldsymbol{x})$ is not monotonic in x_i.
Directional derivatives	Red	$\boldsymbol{dd}_v^-$.
	Blue	$\boldsymbol{dd}_v^+$.
	Black	$0 \in \boldsymbol{dd}_v$.
Point/Vertex	Yellow	Evaluated in sBB and pBB
	Red	Incumbent
	Blue	Infeasible in iBBLC
	Dark Green	Feasible in iBBLC
	Light green	Feasible inherited from feasible box in iBBLC
	Black	Undetermined in iBBLC

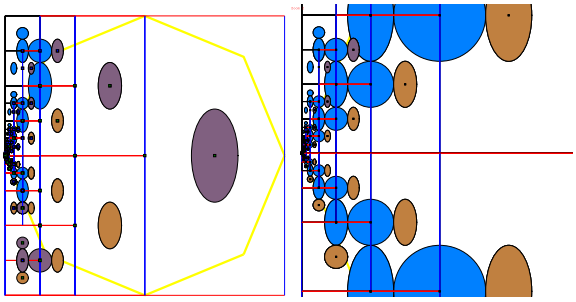

Fig. 9. Algorithm iBBLC on instance 2Pol. Right graph is a zoom over the solution. Graphs show only x_1, x_2 coordinates. So, there is overlapped information about boxes.

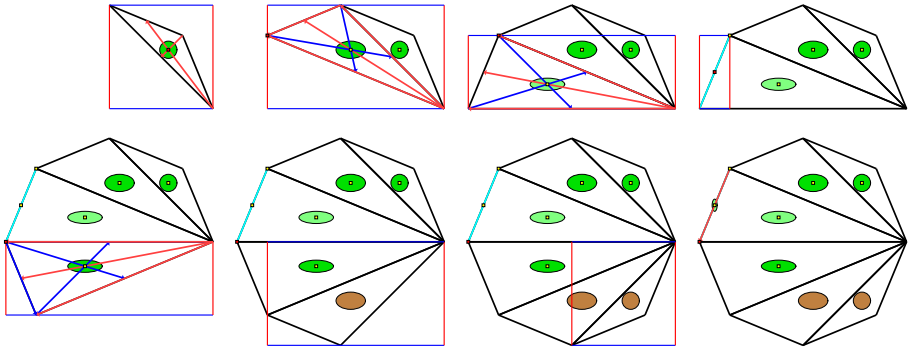

Fig. 10. Algorithm sBB on a 2Pol simplicial partition. Graphs only show x_1, x_2 coordinates. Only monotonic directional derivatives $\boldsymbol{dd}_v$ are drawn.

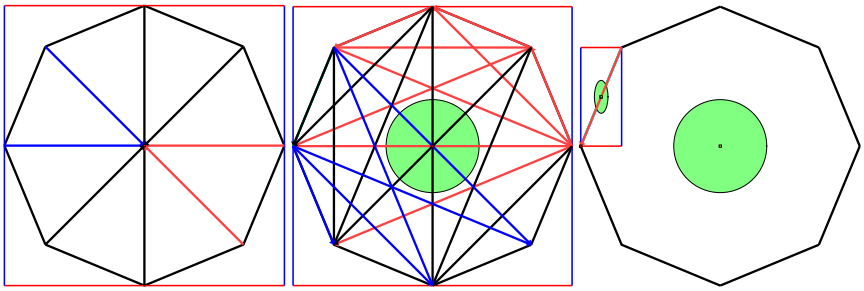

Fig. 11. Algorithm pBB on instance 2Pol. Graphs only show x_1, x_2 coordinates. The left graph corresponds to vertex to centre directional derivatives $\boldsymbol{dd}_v$. The graph in the middle corresponds to a vertex to vertex $\boldsymbol{dd}_v$. In the right graph, the edge-facet is reduced to the solution vertex, i.e. the leftmost vertex.

Figure 10 shows the steps of the sBB algorithm on the simplicial partition of the 2Pol instance. Consider the graphs in a row-wise order, where in each graph the simplex and its interval hull are drawn. In the first graph, the simplex is rejected by a negative directional derivative $\boldsymbol{dd}_v^-$ to a non-border facet. In the second graph, the next simplex is reduced to the left vertex, because a negative directional derivative $\boldsymbol{dd}_v^-$ was found for the other two vertices. In the third graph, a simplex is reduced to the border edge on the left, because there exists a $\boldsymbol{dd}_v^-$ from the vertex not belonging to the border edge. In the fourth graph, the reduced edge is evaluated and stored. In the fifth graph (second row), the next selected simplex is reduced to the vertex solution v^*, the leftmost vertex. Notice the $\boldsymbol{dd}_{v^*}^+$ is negative in the other direction. In the sixth and seventh graphs, selected and evaluated simplices were rejected by the RangeUp test. In the last graph, the stored border edge is selected and reduced to v^* which was already evaluated. Reduced and evaluated simplices are not directly divided, but checked for additional reduction first. This reduces the number of simplex evaluations.

Although we use $\square s$ to get bounds of the objective over the simplex, which incurs in overestimation of the volume of the simplex, the search is restricted to the feasible area. This avoids the feasibility test as is done in iBBLC and its accuracy issue.

Figure 11 illustrates the application of algorithm pBB to the 2Pol instance. Consider the vertices clock-wise, starting at v_1 at the rightmost vertex. The left graph corresponds to two negative directional derivatives $\boldsymbol{dd}_v^-$ in red from vertices v_1 and v_2 to the centre. This implies removing the interior of 2Pol and the facets containing these vertices. We can use more than one vertex, because we are in a 2-polytope, see Fig. 2. In the middle graph, vertices v_1, v_7 and v_8 have a negative vertex to vertex direction $\boldsymbol{dd}_v^-$ shown in red. Direction v_5^* to v_4 results in $\boldsymbol{dd}_v^+$, which is negative in the opposite direction. After checking all directional derivatives, the initial polytope is reduced to edge (v_5, v_6), see Section 4.5. The right graph illustrates the reduction of that edge to the solution vertex v_5^*. For this instance, pBB does not perform a division, only reductions. Therefore, the number of evaluated polytopes is just two and the number of evaluated points is just three.

For the full dimensional 3Pol case in Table 4, we use five different objective functions generating five instances, see Appendix A.2 and Fig. 15 in Appendix A, given by:

3PolIn: The minimum is interior.
3Pol$_{f_2}$: The minimum is in the relative interior of facet f_2 in the (x_1, x_2) plane.
3Pol$_{f_3}$: The minimum is in the relative interior of facet f_3 not parallel to coordinates.
3Pol$_{e_8}$: The minimum is in edge e_8 in the (x_1, x_2) plane, but not parallel to any axis.
3Pol$_{v_6}$: The minimum is at vertex v_6, the top one.

The aim of this experimental design is to investigate the impact of the location of the minimum point on the generation of undetermined boxes in algorithm iBBLC and to observe the impact on pBB. The number of iBBLC boxes is expected to be large when the minimum is at the boundary face not parallel to axes or the face is not parallel to coordinate planes. This number is expected to decrease when the face is parallel to axes. In that case, iBBLC may perform reductions at a given stage of the algorithm. This effect is less when the face is in a coordinate plane and its facets are not parallel to axes, because iBBLC is still generating reduced undetermined boxes. Having an interior minimum is interesting for iBBLC, because it is mostly working on feasible boxes around the minimum.

These expectations are reflected in the results reported in Table 4. Algorithm iBBLC can solve instances 3PolIn, 3Pol$_{f_2}$, 3Pol$_{e_8}$ and 3Pol$_{v_6}$, but it cannot solve instance 3Pol$_{f_3}$ in less than 5 min. Selecting the first widest component to divide, iBB-LC was able to solve instance 3Pol$_{v_6}$. Selecting the last widest component to divide, it could not solve the problem in less than 5 min, because the box centres around the minimum are not feasible.

The advantage of algorithms sBB and pBB is that they do not have to deal with infeasible or undetermined sets; sBB works directly with the simplicial partition (no constraints). Additionally, sBB and pBB can reduce to any border face. Actually, sBB is better than iBBLC for all 3Pol instances apart from 3PolIn and pBB is the best apart from the measure of the number of generated points in the 3PolIn instance.

Instances Ex62 and Ex62In (see Apendix A.4.1 and A.4.2) have a full dimensional 2-simplex feasible set where the minimum point is at an edge and is interior, respectively. Results are provided in Table 4. They show that Ex62 requires in iBBLC two order of magnitude more boxes than simplices in sBB and polytopes in pBB. This is due to the ability of sBB and pBB to reduce to the edge where the minimum is, as illustrated in Fig. 12. Comparing pBB and sBB, pBB has the advantage of reducing the volume overestimation of the inclusion of the polytopes p^ℓ and p^r, resulting in the division of the widest coordinate of p, by $\square p^\ell$ and $\square p^r$. The less tight interval hull is caused by having fewer axis oriented subsimplices after a longest edge bisection of a simplex.

Figure 13 shows the graphical output of the three algorithms for the instance Ex62In. Having the minimum inside the feasible area, iBBLC generates more full dimensional feasible boxes than with the minimum at the boundary. Algorithm iBBLC performs better than sBB, because more bisections are needed to reduce a simplex to half size (length of the longest edge) than for a box with the same size. This difference increases with the dimension. Algorithm pBB generates the smallest number of sets, but the number of generated points increases mainly due to the number of new vertices in the bisection.

Summarizing, pBB gives the best numbers for the performance measure for all cases in Table 4 apart from the number of generated points (not all of them evaluated) in the instances 3PolIn and Ex62In.

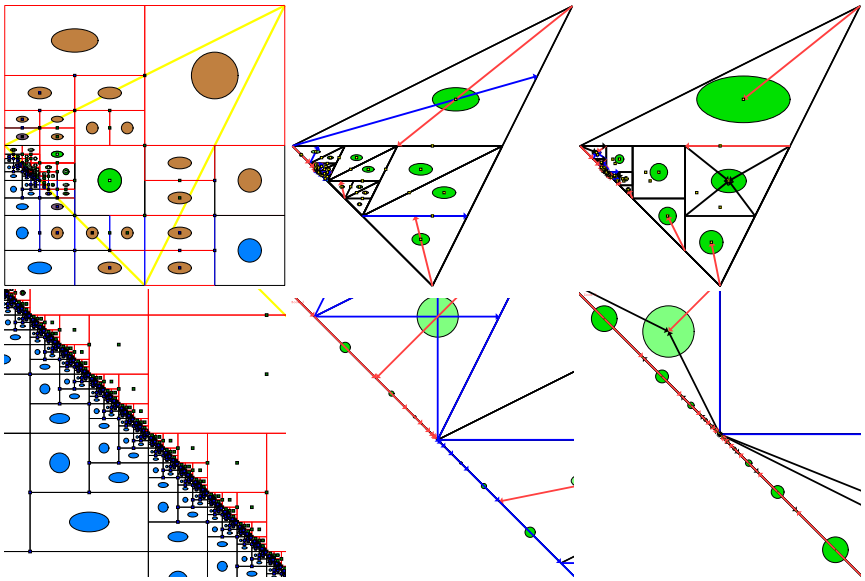

Fig. 12. Graphical output of algorithms iBBLC (left), sBB (centre) and pBB (right) on instance Ex62. The bottom row zooms in on the minimum point.

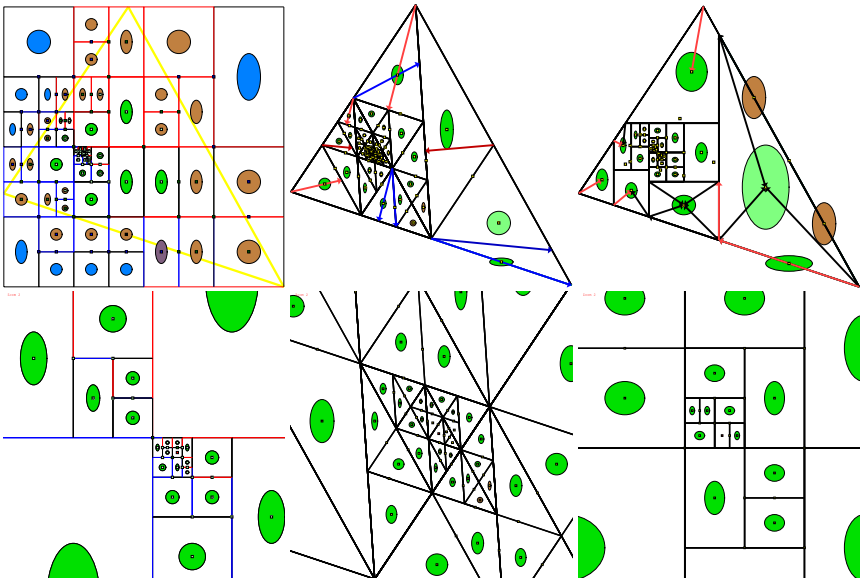

Fig. 13. Graphical output of algorithms iBBLC (left), sBB (centre) and pBB (right) on Ex62In instance. The bottom row zooms in on the minimum point.

6. Conclusions

In this contribution, we developed a new spatial branch and bound approach based on polytope subsets specifically for solving problems with a polytope feasible set. We derived various theoretical properties which can be exploited. We compared the behaviour with two other approaches, namely Interval Arithmetic branch and bound using linear constraints (iBBLC) and simplicial branch and bound (sBB).

It is typical for sBB to face several challenges i) it requires an initial partition of the feasible set into simplices, ii) when using the interval hull, used by Interval Arithmetic to obtain bounds of the real range of the objective function and its derivatives, the interval hull may lead to a less tight enclosure of a simplex and therefore overestimates the simplex volume, iii) branching a simplex to its half requires in general more divisions than box divisions. A clear advantage is that the linear constraints and the corresponding precision are not required, as the convex hull of vertices guarantees generating feasible solutions.

We found that iBBLC has a problem to deal with linear constraints. The precision of the linear constraints plays a role in the speed of convergence of the algorithm. The main drawback of iBBLC is that it generates many undetermined boxes for feasible regions with the minimum at the boundary, when this boundary is not parallel to axes. This presents issues when having to deal with a polytope feasible set with a polytope dimension less than that of the objective function which is not parallel to axes. For instances where the minimum point is in the interior of the feasible region and the dimension is large, iBBLC in general performs better.

The main finding is that the pBB algorithm performs better than sBB and iBBLC for feasible regions where the minimum point is at a boundary which is not a face parallel to the axes. One advantage is that it does not have to deal with linear constraints and the corresponding precision issue. Moreover, when using a coordinate wise bisection, the division promotes box shaped polytopes, reducing the overestimation of the volume of the polytope by its interval hull.

The main challenge of pBB is the need to keep track of vertices, edges and facets of the partition sets and the cost of the division by bisecting the widest coordinates of the interval hull of a polytope, as was shown in Algorithm 2. This can be computationally expensive for large dimensional instances. We are checking the impact on larger dimensional instances which we need to generate, as no benchmark GO over a polytope exists yet. We aim at a future presentation of the benchmarks and results in a forthcoming paper. In future research, we plan to improve and parallelize the code.

A. Description of Test Instances

We provide for each instance the objective function, the vertex set W and corresponding inequalities $Ax \leqslant b$. To reproduce numerical results, the small test instances are: three polytope feasible sets called 2Pol, 3Pol and 4Pol and two 2-dimensional simplicial sets. The instances facilitate experimental variation to have the optimum in a vertex, edge or

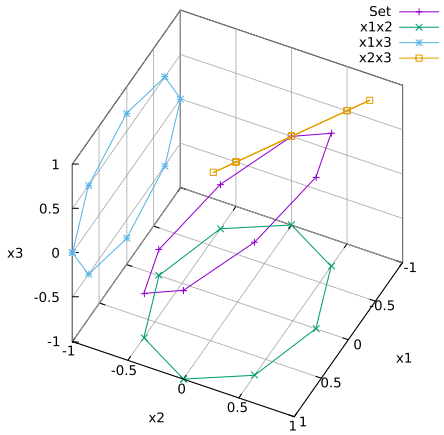

Fig. 14. 2Pol and its projection to $x_i x_j$ planes.

face and to vary the polytope dimension of the set. The polytope branch and bound algorithm requires as input the set of vertices, the set of edges as an Edge-Vertex relation and the facets given by a Facet-Edge relation. In this appendix, we will provide these relations in order to facilitate reproduction of results. Edges and facets of a simplex are easily determined from its vertices: all vertices are connected and a facet is obtained by removing a vertex.

A.1. *2Pol: 2-polytope, $n = 3$, 8 vertices, 8 edges that also are 1-facets*

$$f(x) = 12 - (x_2 - 2)(x_1 - x_3 + 2) - (x_1 - 1)(x_3 - 3), \quad x^* = (-1, 0, 0)^T, \quad f^* = 8$$

$$W^T = \begin{pmatrix} 1 & 0 & 0 \\ 0.707106781 & 0.5 & 0.5 \\ 0 & 0.707106781 & 0.707106781 \\ -0.707106781 & 0.5 & 0.5 \\ -1 & 0 & 0 \\ -0.707106781 & -0.5 & -0.5 \\ 0 & -0.707106781 & -0.707106781 \\ 0.707106781 & -0.5 & -0.5 \end{pmatrix}$$

Edge-Vertex: $\begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 5 & 6 \\ 6 & 7 \\ 7 & 8 \\ 8 & 1 \end{pmatrix}$ Facet-Edge $(1, 2, 3, 4, 5, 6, 7, 8)$

$$A = \begin{pmatrix} 0 & 1 & -1 \\ 0 & -1 & 1 \\ 1 & 0.5857864376 & 0 \\ 1 & -0.5857864376 & 0 \\ 0.4142135624 & 1.4142135623 & 0 \\ -0.4142135624 & 1.4142135623 & 0 \\ -1 & 0.5857864376 & 0 \\ -1 & -0.5857864376 & 0 \\ -0.4142135624 & -1.4142135623 & 0 \\ 0.4142135624 & -1.4142135623 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

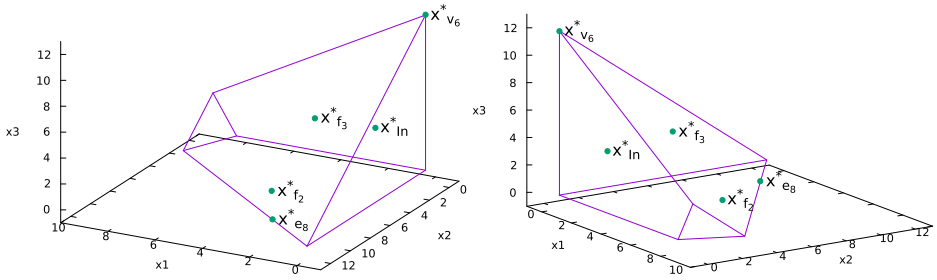

Fig. 15. 3Pol instance.

A.2. *3Pol: 3-polytope, $n = 3$, 6 vertices, 9 edges, and 5 2-facets*

$f_{In}(x) = (x_1 - 1)(x1 + 2x_2 - 5) + 2(x_2 - 2)^2 + (x_3 - 3)(3x_3 + 2x_2 - 15)$, $x^* = (1.5, 1.5, 3.5)^T$, $f^* = -0.5$

$f_{f_2}(x) = (x_1 - 4)^2 + 2(x2 - 3)^2 + 2(x2 - 3)(x3 - 6) - (x_3 - 6)^2$, $x^* = (4, 6, 0)^T$, $f^* = -54$

$f_{f_3}(x) = 12 - (x_1 - 1)(x_2 + x_3 - 5) - (x_2 - 2)(x_3 - 3)$, $x^* = (3, 4, 5)^T$, $f^* = 0$

$f_{e_8}(x) = 12 - (x_2 - 2)(x_1 - x_3 + 2) - (x_1 - 1)(x_3 - 3)$, $x^* = (2.5, 9.5, 0.0)^T$, $f^* = -17.25$

$f_{v_6}(x) = (x_1 - 4)^2 + 2(x2 - 3)^2 + 2(x2 - 3)(x3 - 5) - (x_3 - 5)^2$, $x^* = (0, 0, 12)^T$, $f* = -57$

$$W^T = \begin{pmatrix} 0 & 0 & 0 \\ 8 & 0 & 0 \\ 9 & 0 & 3 \\ 9 & 3 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 12 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 1 & 1 \\ 3 & -1 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 12 \\ 24 \end{pmatrix}.$$

Edge-Vertex: $\begin{pmatrix} 1 & 2 \\ 1 & 5 \\ 1 & 6 \\ 2 & 3 \\ 2 & 4 \\ 3 & 4 \\ 3 & 6 \\ 4 & 5 \\ 5 & 6 \end{pmatrix}$ Facet-Edge: $\begin{pmatrix} 2 & 3 & 9 \\ 1 & 2 & 5 & 8 \\ 6 & 7 & 8 & 9 \\ 1 & 3 & 4 & 7 \\ 4 & 5 & 6 \end{pmatrix}$

A.3. *4Pol: 4-polytope, $n = 6$, 8 vertices 24 edges and 16 facets*

$f(x) = \frac{1}{2} \sum_{i=1}^{6} (x_i - t_i)^2$ with $t = (0.2, 0.8, 0.8, 0.3, 0.8, 0.8)$.

$x^* = (0.13768563491, 0.68875427255, 0.80067434925, 0.25519460047, 0.81977663242, 0.60167627283)^T$, $f^* = 0.028995043249$

$$W^T = \begin{pmatrix} 1.00 & 0.67 & 0.46 & 1.00 & 0.31 & 0.11 \\ 0.86 & 1.00 & 0.80 & 1.00 & 0.58 & 0.00 \\ 0.33 & 1.00 & 1.00 & 0.54 & 0.89 & 0.31 \\ 0.00 & 0.86 & 1.00 & 0.20 & 1.00 & 0.58 \\ 0.00 & 0.33 & 0.54 & 0.00 & 0.69 & 0.89 \\ 0.14 & 0.00 & 0.20 & 0.00 & 0.42 & 1.00 \\ 0.67 & 0.00 & 0.00 & 0.46 & 0.11 & 0.69 \\ 1.00 & 0.14 & 0.00 & 0.80 & 0.00 & 0.42 \end{pmatrix}$$

Edge-Vertex is $E = \{(1, 2), (1, 3), (1, 4), (1, 6), (1, 7), (1, 8), (2, 3), (2, 4), (2, 5), (2, 7), (2, 8),$
$(3, 4), (3, 5), (3, 6), (3, 8), (4, 5), (4, 6), (4, 7), (5, 6), (5, 7), (5, 8), (6, 7), (6, 8), (7, 8)\}$

$$\text{Facet-Vertex}=\begin{pmatrix}
5 & 6 & 7 & 8\\
4 & 5 & 6 & 7\\
3 & 4 & 6 & 8\\
3 & 4 & 5 & 6\\
2 & 5 & 7 & 8\\
2 & 4 & 5 & 7\\
2 & 3 & 5 & 8\\
2 & 3 & 4 & 5\\
1 & 6 & 7 & 8\\
1 & 4 & 6 & 7\\
1 & 3 & 6 & 8\\
1 & 3 & 4 & 6\\
1 & 2 & 7 & 8\\
1 & 2 & 4 & 7\\
1 & 2 & 3 & 8\\
1 & 2 & 3 & 4
\end{pmatrix}
\quad
\text{Facet-Edge}=\begin{pmatrix}
19 & 20 & 21 & 22 & 23 & 24\\
16 & 17 & 18 & 19 & 20 & 22\\
13 & 14 & 15 & 19 & 21 & 23\\
12 & 13 & 14 & 16 & 17 & 19\\
9 & 10 & 11 & 20 & 21 & 24\\
8 & 9 & 10 & 16 & 18 & 20\\
7 & 9 & 11 & 13 & 15 & 21\\
7 & 8 & 9 & 12 & 13 & 16\\
4 & 5 & 6 & 22 & 23 & 24\\
3 & 4 & 5 & 17 & 18 & 22\\
2 & 4 & 6 & 14 & 15 & 23\\
2 & 3 & 4 & 12 & 14 & 17\\
1 & 5 & 6 & 10 & 11 & 24\\
1 & 3 & 5 & 8 & 10 & 18\\
1 & 2 & 6 & 7 & 11 & 15\\
1 & 2 & 3 & 7 & 8 & 12
\end{pmatrix}$$

$$A=\begin{pmatrix}
1.0346255075 & 1.8769125139 & -1.5028759229 & -0.8644311092 & 0.5173127538 & 0.9384562569\\
-1.0346255075 & -1.8769125139 & 1.5028759229 & 0.8644311092 & -0.5173127538 & -0.9384562569\\
6.0062575069 & -3.3108773986 & 2.7662428592 & -4.8093130218 & 3.0031287535 & -1.6554386993\\
-6.0062575069 & 3.3108773986 & -2.7662428592 & 4.8093130218 & -3.0031287535 & 1.6554386993\\
-2.8340080972 & 0 & 0 & 2.2862586330 & -1.4646344368 & 0.0119075970\\
-0.6944444444 & 0 & -0.3472222222 & 0 & -0.2240143369 & -0.7392473118\\
0.5476040649 & 0.5060986223 & 0 & 0 & 0.0354804590 & 0.9084336382\\
0.6872114249 & 0.1109560113 & 0 & 0 & 0.5028812771 & 0.6925802641\\
-1.2208907981 & 1.3210167630 & 0 & 0 & -2.1914666839 & 0.0855915507\\
-0.7656573715 & 0.0797827849 & 0 & 0 & -0.7264093886 & -0.5900065628\\
1.9394316532 & 0 & 0.9800871189 & -1.4519809168 & 0 & 0.5289359054\\
0 & 0 & 1.7750801649 & -1.0650480989 & -2.3362345396 & -0.3893724232\\
0 & 0 & -1.7478574650 & 1.0487144790 & 2.3004059540 & 0.3834009923\\
-1.9462947543 & 0 & -0.9835553705 & 1.4571190674 & 0 & -0.5308076603\\
0.7639076121 & -0.0796004571 & 0 & 0 & 0.7247493228 & 0.5886582187\\
1.2139118148 & -1.3134654292 & 0 & 0 & 2.1789395934 & -0.0851022833\\
-0.6916177371 & -0.1116674471 & 0 & 0 & -0.5061056878 & -0.6970210007\\
-0.5489122411 & -0.5073076459 & 0 & 0 & -0.0355652185 & -0.9106038034\\
0.6910387873 & 0 & 0.3455193937 & 0 & 0.2229157379 & 0.7356219349\\
-1.8534119629 & -1.0362257793 & 0 & 1.6301600674 & 0 & -0.7497893850
\end{pmatrix}$$

$b^T = (1, -1, 1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1, -1, 1, -1).$

A.4. *Ex62:* $2-$*simplex and* $n = 2$

From Karhbet and Kearfott (2017):

$f(x) = \frac{1}{4}x_1^2 + x_1 + x_2 + \frac{1}{4}x_1x_2 + 0.5x_2^2$.

A.4.1. *Ex62: min at edge*

$x^* = (-1.625, -0.5625)^T$, $f^* = -1.140625$

$$W^T = \begin{pmatrix} -2 & 0\\ 2 & -3\\ 0 & 3 \end{pmatrix}, \quad A = \begin{pmatrix} 3 & 1\\ -3 & 2\\ -3 & 4 \end{pmatrix}, \quad b = \begin{pmatrix} 3\\ 6\\ 6 \end{pmatrix}.$$

A.4.2. *Ex62In: min is interior*

$x^* = (-1.71428572387, -0.57142855227)^T$, $f^* = -1.1428571429$

$$W^T = \begin{pmatrix} -3 & -1\\ -1 & 1\\ 1.5 & -2 \end{pmatrix}, \quad A = \begin{pmatrix} -1 & 1\\ 6 & 5\\ -2 & -9 \end{pmatrix}, \quad b = \begin{pmatrix} 2\\ -1\\ 15 \end{pmatrix}.$$

Funding

This work has been funded by Grant PID2021-123278OB-I00 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe".

References

Assarf, B., Gawrilow, E., Herr, K., Joswig, M., Lorenz, B., Paffenholz, A., Rehn, T. (2017). Computing convex hulls and counting integer points with `polymake`. *Mathematical Programming Computation* , 9(1), 1–38. https://doi.org/10.1007/s12532-016-0104-z.

Casado, L.G., G.-Tóth, B., Messine, F., Hendrix, E.M.T. (2021). Directional derivative bounds and border facets in simplicial B&B monotonicity tests. In: *SCAN'20: The 19th International Symposium on Scientific Computing, Computer Airthmetic and Verified Numerical Computations*, Szeged, Hungary, pp. 18–19.

Casado, L.G., G.-Tóth, B., Hendrix, E.M.T., Messine, F. (2022). On monotonicity detection in simplicial branch and bound over a simplex. In: *Computational Science and Its Applications – ICCSA 2022 Workshops*. Springer International Publishing, Cham, pp. 113–126. 978-3-031-10562-3.

Casado, L.G., G.-Tóth, B., Hendrix, E.M.T., Messine, F. (2025). Local search versus linear programming to detect monotonicity in simplicial branch and bound. *Journal of Global Optimization*, 91, 311–330.

Fernández, J., G.-Tóth, B. (2022). Interval tools in branch-and-bound methods for global optimization. In: *The Palgrave Handbook of Operations Research*. Springer, pp. 237–267. https://doi.org/10.1007/978-3-030-96935-6_8.

G.-Tóth, B., Casado, L.G., Hendrix, E.M.T., Messine, F. (2021). On new methods to construct lower bounds in simplicial branch and bound based on interval arithmetic. *Journal of Global Optimization*, 80(4), 779–804.

G.-Tóth, B., Hendrix, E.M.T., Casado, L.G., Messine, F. (2024). On dealing with minima at the border of a simplicial feasible area in simplicial branch and bound. *Journal of Optimization Theory and Applications*, 203, 1794–1819. https://doi.org/10.1007/s10957-024-02480-9.

Gencsi, M., G.-Tóth, B. (2025). Efficient use of optimality conditions in Interval Branch and Bound methods. *EURO Journal on Computational Optimization*, 13, 100108. https://doi.org/10.1016/j.ejco.2025.100108.

Hansen, E.R., Walster, G.W. (2004). *Global Optimization Using Interval Analysis*, 2nd ed. Marcel Dekker Inc., New York.

Hendrix, E.M.T., G.-Tóth, B. (2010). *Introduction to Nonlinear and Global Optimization*. Springer, New York.

Hendrix, E.M.T., Casado, L.G., G.-Tóth, B., Messine, F. (2024). On Polytopal Branch and Bound with Monotonicity. In: Gervasi, O., Murgante, B., Garau, C., Taniar, D., C. Rocha, A.M.A., Faginas Lago, M.N. (Eds.), *Computational Science and Its Applications – ICCSA 2024 Workshops*. Springer Nature Switzerland, Cham, pp. 397–414. 978-3-031-65223-3.

Karhbet, S.D., Kearfott, R.B. (2017). Range Bounds of Functions over Simplices, for Branch and Bound Algorithms. *Reliable Computing*, 25, 53–73. https://interval.louisiana.edu/reliable-computing-journal/volume-25/reliable-computing-25-pp-053-073.pdf.

Kearfott, R.B. (1992). An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization*, 2(3), 259–280. https://doi.org/10.1007/BF00171829.

López, F.J., Duval, A.M. (2012). Algorithms to determine the edges of a convex hull from its vertices. *International Journal of Mathematical Modelling and Numerical Optimisation*, 3(3), 184–209. https://doi.org/10.1504/IJMMNO.2012.047704.

Moore, R.E., Kearfott, R.B., Cloud, M.J. (2009). *Introduction to interval analysis*. Society for Industrial and Applied Mathematics, USA. 0898716691. https://doi.org/10.1137/1.9780898717716.

Paulavičius, R., Žilinskas, J. (2014). *Simplicial Global Optimization*. Springer, New York. 978-1-4614-9092-0.

Rall, L.B. (Ed.) (1981). *Examples of Software for Automatic Differentiation and Generation of Taylor Coefficients*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 54–90. 978-3-540-38776-3.

L.G. Casado has a PhD from University of Málaga. He is full professor at the University of Almería. His research activities include exhaustive search in global optimization algorithms, parallel computing, network security, etc. See https://sites.google.com/ual.es/leo.

B.G.-Tóth is a senior researcher at University of Szeged. Her research interests are rigorous global optimization methods, and their application to facility location problems. She obtained her PhD from the University of Almería.

E.M.T. Hendrix is a full professor at the Universidad de Málaga. His research interests are global and dynamic optimization and computational impacts. He obtained his PhD from Wageningen University and his MSc and Bsc from Tilburg University.

F. Messine is a full professor at the University of Toulouse, specifically at ENSEEIHT and the LAPLACE-CNRS laboratory. His research interests focuses on deterministic global optimization, topology and shape optimization and their applications to optimize the designs of electromechanical actuators such as electrical machines and space thrusters. He obtained his PhD and his Habilitation (accreditation to supervise research) at Toulouse-INP University.