Journal of Global Optimization (2021) 80:779–804
https://doi.org/10.1007/s10898-021-01053-8

On new methods to construct lower bounds in simplicial branch and bound based on interval arithmetic

B. G.-Tóth[1] · L. G. Casado[2] · E. M. T. Hendrix[3] · F. Messine[4]

Received: 1 December 2020 / Accepted: 8 May 2021 / Published online: 10 July 2021

Abstract

Branch and Bound (B&B) algorithms in Global Optimization are used to perform an exhaustive search over the feasible area. One choice is to use simplicial partition sets. Obtaining sharp and cheap bounds of the objective function over a simplex is very important in the construction of efficient Global Optimization B&B algorithms. Although enclosing a simplex in a box implies an overestimation, boxes are more natural when dealing with individual coordinate bounds, and bounding ranges with Interval Arithmetic (IA) is computationally cheap. This paper introduces several linear relaxations using gradient information and Affine Arithmetic and experimentally studies their efficiency compared to traditional lower bounds obtained by natural and centered IA forms and their adaption to simplices. A Global Optimization B&B algorithm with monotonicity test over a simplex is used to compare their efficiency over a set of low dimensional test problems with instances that either have a box constrained search region or where the feasible set is a simplex. Numerical results show that it is possible to obtain tight lower bounds over simplicial subsets.

Keywords Simplex · Branch and bound · Interval arithmetic · Affine arithmetic · Linear programming

This work has been funded by grant RTI2018-095993-B-I00 from the Spanish Ministry.

✉ E. M. T. Hendrix
eligius.hendrix@wur.nl

B. G.-Tóth
boglarka@inf.szte.hu

L. G. Casado
leo@ual.es

F. Messine
frederic.messine@laplace.univ-tlse.fr

1 Department of Computational Optimization, University of Szeged, Szeged, Hungary

2 Informatics Department, University of Almería, CeiA3, Almería, Spain

3 Universidad de Málaga and Wageningen University, Wageningen, Netherlands

4 LAPLACE-ENSEEIHT, Toulouse-INP, University of Toulouse, Toulouse, France

1 Introduction

A review of simplicial Branch and Bound (B&B) can be found in [15]. Recently, there is a renewed interest in generating tight bounds over simplicial partition sets. Karhbet and Kearfott [7] discuss the idea of using range computation over simplices based on Interval Arithmetic. In [12], focus is on using second derivative enclosures for generating bounds. These works do not take monotonicity considerations over the simplex into account as discussed by [6]. Our research question is how information on the bounds of first derivatives can be used to derive tight bounds and to create new monotonicity tests in simplicial B&B. To investigate this question, we derive bounds based on derivative information and implement them in a B&B algorithm to compare the different techniques.

The rest of this paper is organized as follows. Section 2 introduces the notation. Section 3 presents several approaches to obtain lower bounds of a function over a simplex. Section 4 deals with monotonicity over a simplex. Section 5 describes the Global Optimization B&B algorithm to compare lower bounding methods over a simplex. Section 6 compares the results of the bounding techniques numerically on a large number of instances. Finally, Sect. 7 presents our findings.

2 Preliminaries

Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ which has to be minimized over a feasible set $D \subset \mathbb{R}^n$, which is either a box or a simplex, on which f is differentiable:

$$\min_{x \in D} f(x).$$

The simplicial B&B algorithm to be investigated uses simplicial partition sets S and lower bounds of $\min_{x \in S} f(x)$.

Notation 1 *Let $\mathcal{V} = \{v_0, \ldots, v_n\} \subset \mathbb{R}^n$ denote a set of $n+1$ affinely independent vertices. For the component i of vertex j, we use the notation $(v_j)_i$.*

Notation 2 *An n-simplex S is determined by the convex hull of $\mathcal{V}$, i.e. $S = \mathrm{conv}(\mathcal{V})$*

$$S = \left\{ y = \sum_{j=0}^{n} \lambda_j v_j \mid \lambda_j \geq 0, j = 0, \ldots, n, \sum_{j=0}^{n} \lambda_j = 1 \right\}. \tag{1}$$

Notation 3 *We denote intervals by boldface letters and their lower and upper bound by 'underline' and 'overline', respectively. The radius of an interval $\boldsymbol{x} = [\underline{x}, \overline{x}]$ is denoted by $\mathrm{rad}\,(\boldsymbol{x}) = \frac{\overline{x}-\underline{x}}{2}$ and its midpoint by $\mathrm{mid}\,(\boldsymbol{x}) = \frac{\overline{x}+\underline{x}}{2}$. For an interval vector (also called a box) these are taken component-wise. The width of a box $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ is to be understood as $\mathrm{wid}\,(\boldsymbol{x}) = 2 \max_{i=1,\ldots,n} \mathrm{rad}\,(\boldsymbol{x}_i)$.*

Notation 4 *The interval hull of a simplex S is denoted by $\square S = \square\mathrm{conv}(\mathcal{V})$, that is the smallest interval box enclosing the simplex S. Let $\boldsymbol{x} = \square S$, where*

$$\boldsymbol{x}_i = [\underline{x}_i, \overline{x}_i] = [\min_{v \in \mathcal{V}} (v)_i, \max_{v \in \mathcal{V}} (v)_i] \qquad \forall i \in \{1, \ldots, n\}. \tag{2}$$

Remark 1 For cases where $\boldsymbol{x} = \square S \subseteq D$, f is differentiable over $\boldsymbol{x}$. Notice that if D is a box and $S \subset D$, automatically we have $\square S \subseteq D$. However, if D is a simplex, then f is not necessarily differentiable over $\boldsymbol{x} = \square S$.

Notation 5 *The boundary and interior of set S is denoted by ∂S and* int S*, respectively, where $S = \partial S \cup \text{int } S$ and $\partial S \cap \text{int } S = \emptyset$.*

3 Bounding techniques over a simplex

3.1 Extension of standard interval bounding techniques to simplices

Extensive investigation on Interval Arithmetic has lead to many ways to derive rigorous bounds, see for instance [5,8,13,17].

Notation 6 *Let $\boldsymbol{f}$ denote the natural interval extension [13] of an expression f with*

$$\boldsymbol{f}(\boldsymbol{x}) = [\underline{\boldsymbol{f}}(\boldsymbol{x}), \overline{\boldsymbol{f}}(\boldsymbol{x})] \supseteq [\min_{x \in \boldsymbol{x}} f(x), \max_{x \in \boldsymbol{x}} f(x)], \ \forall \boldsymbol{x} \subseteq D.$$

Remark 2 $\forall x \in S \subset \boldsymbol{x} = \Box S,\ f(x) \in \boldsymbol{f}(\boldsymbol{x})$.

Notation 7 *Let $\nabla \boldsymbol{f}(\boldsymbol{x})$ denote an enclosure of the gradient and $\nabla \boldsymbol{f}_i(\boldsymbol{x}) = [\underline{\nabla \boldsymbol{f}_i}(\boldsymbol{x}), \overline{\nabla \boldsymbol{f}_i}(\boldsymbol{x})]$ the i-th component of the interval gradient. They can be computed using Interval Arithmetic[1] and Automatic Differentiation[2] [16].*

Remark 3 $\forall x \in \boldsymbol{x},\ \frac{\partial f}{\partial x_i}(x) \in \nabla \boldsymbol{f}_i(\boldsymbol{x})$. Then, $\forall x \in S \subset \boldsymbol{x} = \Box S,\ \frac{\partial f}{\partial x_i}(x) \in \nabla \boldsymbol{f}_i(\boldsymbol{x})$.

Notation 8 *A centered form on a box $\boldsymbol{x}$ with center c is denoted by $\boldsymbol{f}_c(\boldsymbol{x})$. It is in fact the interval extension of the first-order Taylor expansion using $\nabla \boldsymbol{f}(\boldsymbol{x})$:*

$$\boldsymbol{f}_c(\boldsymbol{x}) = \boldsymbol{f}(c) + (\boldsymbol{x} - c)^T \nabla \boldsymbol{f}(\boldsymbol{x}), \text{ with } c \in \boldsymbol{x}.$$

Usually c is the midpoint (or center) of box $\boldsymbol{x}$. In that case, we refer to $\boldsymbol{f}_{cb}(\boldsymbol{x})$ where $cb = \text{mid}(\boldsymbol{x})$. The lower bound $\underline{\boldsymbol{f}_c}(\boldsymbol{x})$ can also be written as $\underline{\boldsymbol{f}(c) + (\boldsymbol{x} - c)^T \nabla \boldsymbol{f}(\boldsymbol{x})} = \underline{\boldsymbol{f}}(c) + \underline{(\boldsymbol{x} - c)^T \nabla \boldsymbol{f}(\boldsymbol{x})}$, where underline takes the lower bound of the formula computed by IA.

Remark 4 $\forall x \in S \subset \boldsymbol{x} = \Box S,\ f(x) \in \boldsymbol{f}_c(\boldsymbol{x})$. Thus, $\boldsymbol{f}_c(\boldsymbol{x})$ provides lower and upper bounds of f over S, even if $c \notin S$.

Baumann [3] proposed another base-point instead of the center cb to improve the lower and upper bounds of the centered form.

Notation 9 *We denote the Baumann base-point for the optimal lower bound in the centered form on a box by bb^-. Component i is given by*

$$bb_i^- = \begin{cases} \dfrac{\underline{x}_i \overline{\nabla \boldsymbol{f}}_i(\boldsymbol{x}) - \overline{x}_i \underline{\nabla \boldsymbol{f}_i}(\boldsymbol{x})}{\text{wid}\,(\nabla \boldsymbol{f}_i(\boldsymbol{x}))} & \text{if } 0 \in \nabla \boldsymbol{f}_i(\boldsymbol{x}) \\ \underline{x}_i & \text{if } \underline{\nabla \boldsymbol{f}_i}(\boldsymbol{x}) > 0 \\ \overline{x}_i & \text{if } \overline{\nabla \boldsymbol{f}}_i(\boldsymbol{x}) < 0. \end{cases}$$

Any centered form (with a base-point $y \in \boldsymbol{x}$) can be tightened based on the vertices of simplex S.

[1] cs.utep.edu.

[2] autodiff.org.

Proposition 1 *Let*

$$\underline{\boldsymbol{f}_y}(S) = \underline{\boldsymbol{f}}(y) + \min_{v\in\mathcal{V}}\{\underline{(v-y)^T\nabla\boldsymbol{f}(\boldsymbol{x})}\},\ \ y \in \boldsymbol{x}. \tag{3}$$

Then $\underline{\boldsymbol{f}_y}(S) \le \min_{x\in S} f(x)$.

Proof A first-order Taylor form provides a concave lower bounding function [3,25]. A concave function takes its minimum over a convex set at its extreme points. Consequently, the lower bounding function over the simplex takes its minimum at a vertex of the simplex. Thus, instead of computing the interval enclosure over $\boldsymbol{x}$=$\Box S$, taking the minimum over the simplex vertices provides a valid lower bound. □

Remark 5 We can use $y = cb$ or $y = bb^-$ in (3).

Now, it is interesting to see how the Baumann point bb^- can be generalized to a simplicial base-point. For bb^-, the aim is to select the best base-point for the Taylor form, such that the lower bound is as high as possible. For a simplex, instead of using the limits of enclosing box $\boldsymbol{x}$=$\Box S$, we use the simplex vertices.

The highest lower bound in (3) over a simplex is taken at the base-point

$$\operatorname*{argmax}_{y\in\boldsymbol{x}} \min_{v\in\mathcal{V}} \left(\underline{\boldsymbol{f}}(y) + \underline{(v-y)^T\nabla\boldsymbol{f}(\boldsymbol{x})}\right) = \operatorname*{argmax}_{y\in\boldsymbol{x}} \left(\underline{\boldsymbol{f}}(y) + \min\nolimits_{v\in\mathcal{V}} \underline{(v-y)^T\nabla\boldsymbol{f}(\boldsymbol{x})}\right). \tag{4}$$

Obviously, optimizing (4) is a nonlinear problem as it includes the optimization of $f(y)$ varying y. Therefore, it is advisable to optimize only the second part.

Definition 1 Let us define $bs^- = \operatorname*{argmax}_{y\in\boldsymbol{x}} \min_{v\in\mathcal{V}} \underline{(v-y)^T\nabla\boldsymbol{f}(\boldsymbol{x})}$ as the Baumann point over the simplex. This point can be found by an interval linear program:

$$\begin{aligned} \max_{y\in\boldsymbol{x}, z\in\mathbb{R}} \quad & z \\ \text{s.t.} \quad & z \le \underline{(v-y)^T\nabla\boldsymbol{f}(\boldsymbol{x})},\ \ \forall v \in \mathcal{V}. \end{aligned} \tag{5}$$

Let (z^*, y^*) be the optimum of (5). Then we take base point $bs^- = y^*$ with the corresponding lower bound $\underline{\boldsymbol{f}_{bs^-}}(\boldsymbol{x}) = \boldsymbol{f}(bs^-) + z^*$.

Notation 10 $\nabla^w \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^n$ *denotes gradient bounds with components* $\nabla^w \boldsymbol{f}_i(\boldsymbol{x}) = \underline{\nabla\boldsymbol{f}_i}(\boldsymbol{x})$ *if* $w_i = \underline{x}_i$ *and* $\nabla^w \boldsymbol{f}_i(\boldsymbol{x}) = \overline{\nabla\boldsymbol{f}}_i(\boldsymbol{x})$ *if* $w_i = \overline{x}_i$.

Remark 6 In Notation 10, all possible variations of lower and upper bounds of the gradients are taken into account when considering all vertices w of $\boldsymbol{x}$.

Writing (5) as a linear program requires 2^n constraints for each vertex $v \in \mathcal{V}$:

$$\begin{aligned} \max_{y\in\boldsymbol{x}, z\in\mathbb{R}} \quad & z \\ \text{s.t.} \quad & z \le (v-y)^T\nabla^w\boldsymbol{f}(\boldsymbol{x}),\quad \forall v \in \mathcal{V},\ \forall w \text{ vertex of } \boldsymbol{x}. \end{aligned} \tag{6}$$

The constraints in (6) can be written as 2^n linear inequalities

$$\begin{aligned} \max_{y\in\boldsymbol{x}, z\in\mathbb{R}} \quad & z \\ \text{s.t.} \quad & z + y^T\nabla^w\boldsymbol{f}(\boldsymbol{x}) \le \min_{v\in\mathcal{V}} v^T\nabla^w\boldsymbol{f}(\boldsymbol{x}),\ \forall w \text{ vertex of } \boldsymbol{x}. \end{aligned} \tag{7}$$

Note that we do not force bs^- to be in simplex S, because it may happen that a point outside S would give the best lower bound. In case we want to use $\overline{f}(bs^-)$ to update the upper bound of the global minimum in a B&B algorithm, bs^- has to be in the initial search region. In our experiments we force bs^- to be in S by adding $y \in S$ using simplex inclusion constraints (1) to (7) in a similar way as it is done in (10).

Notice that (5), (6) and (7) are equivalent descriptions of the same problem, thus providing the same optimum corresponding to the same bound.

3.2 Linear relaxation based lower bounds

Following earlier results in interval based B&B [14,20,21], we can now define other lower bounds for simplicial subsets.

3.2.1 Standard linear relaxation of *f* over a box

Let w be a vertex of $\boldsymbol{x} = \Box S$ and consider a first order Taylor expansion

$$\underline{f_w}(x) = \underline{f}(w) + (x - w)^T \nabla^w \boldsymbol{f}(\boldsymbol{x}) \leq f(x) \quad \forall x \in \boldsymbol{x}. \tag{8}$$

Since we have 2^n vertices of $\boldsymbol{x}$, we obtain 2^n inequalities from Eq. (8), see [10] for more details. Consider the linear program

$$\begin{aligned} \min_{x \in \boldsymbol{x}, z \in \mathbb{R}} \quad & z \\ \text{s.t.} \quad & z \geq \underline{f}(w) + (x - w)^T \nabla^w \boldsymbol{f}(\boldsymbol{x}), \ \forall w \text{ vertex of } \boldsymbol{x}. \end{aligned} \tag{9}$$

Let (x^*, z^*) be the optimal solution of (9), then

$$f(x) \geq z^*, \forall x \in \boldsymbol{x},$$

such that z^* is a lower bound of f over $\boldsymbol{x}$. z^* is also a lower bound of f over $S \subset \boldsymbol{x} = \Box S$.

3.2.2 Linear relaxation of *f* over a simplex

We now focus on the bounds of f over simplex $S = \text{conv}(\mathcal{V})$. The earlier bound in (9) is valid for f over $\boldsymbol{x} = \Box S$, such that it is also a bound over the simplex S. However, it is interesting to force $x \in \boldsymbol{x}$ to be inside S, like in (1). Introducing the corresponding linear equations into problem (9) provides linear program

$$\begin{aligned} \min_{\substack{x \in \boldsymbol{x}, z \in \mathbb{R} \\ \lambda \in [0,1]^{n+1}}} \quad & z \\ \text{s.t.} \quad & z \geq \underline{f}(w) + (x - w)^T \nabla^w \boldsymbol{f}(\boldsymbol{x}), \ \forall w \text{ vertex of } \boldsymbol{x} \\ & x = \sum_{j=0}^{n} \lambda_j v_j \\ & \sum_{j=0}^{n} \lambda_j = 1. \end{aligned} \tag{10}$$

Let (x^*, z^*, λ^*) be the solution of (10). Then we have that

$$f(x) \geq z^*, \forall x \in S$$

and therefore, z^* is a lower bound of f over S.

A straightforward idea is to consider the vertices of the simplex instead of the vertices of the enclosing box. Unfortunately, such a formulation leads to a Mixed Integer Programming problem, as the piece-wise linear lower bounding function is neither convex nor concave anymore.

3.3 Bounding technique using Affine Arithmetic

This section describes the use of Affine Arithmetic (see [2,4,9,11,18,22]) to generate a linear underestimation of function f over $\boldsymbol{x}=\Box S$. We add the constraint that the solution has to be inside the simplex $S = \text{conv}(\mathcal{V})$, see (1). This provides a linear program.

First, we focus on the transformation of an interval vector into a vector of affine forms. Second, we describe how the computations are made using Affine Arithmetic to provide linear equations. Third, we sketch how the so-obtained linear equations are used to provide linear underestimations of f over $\boldsymbol{x}$ and then we provide the linear program to find a lower bound of f over the simplex S. Fourth, we show a simple way to solve the linear program.

3.3.1 Conversion into affine forms

The interval vector $\boldsymbol{x}=\Box S$ can be converted to an affine form vector, denoted by $\hat{x}$, as follows

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_i \\ \vdots \\ \boldsymbol{x}_n \end{pmatrix} = \begin{pmatrix} [\underline{x}_1, \overline{x}_1] \\ \vdots \\ [\underline{x}_i, \overline{x}_i] \\ \vdots \\ [\underline{x}_n, \overline{x}_n] \end{pmatrix} \rightarrow \hat{x} = \begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_i \\ \vdots \\ \hat{x}_n \end{pmatrix} = \begin{pmatrix} \text{mid}\,(\boldsymbol{x}_1) + \text{rad}\,(\boldsymbol{x}_1)\epsilon_1 \\ \vdots \\ \text{mid}\,(\boldsymbol{x}_i) + \text{rad}\,(\boldsymbol{x}_i)\epsilon_i \\ \vdots \\ \text{mid}\,(\boldsymbol{x}_n) + \text{rad}\,(\boldsymbol{x}_n)\epsilon_n \end{pmatrix}, \tag{11}$$

where $\epsilon_i \in [-1, 1]$ for all $i \in \{1, \ldots, n\}$. The affine form $\hat{x}$ can be transformed back into an interval by changing ϵ_i to $[-1, 1]$. Moreover, for all $x \in \boldsymbol{x}$, there is exactly one corresponding value for ϵ in the affine description,

$$x = T(\boldsymbol{x}, \epsilon) = \text{mid}\,(\boldsymbol{x}) + \text{rad}\,(\boldsymbol{x})\epsilon,$$

where $\epsilon_i = \frac{x_i - \text{mid}\,(\boldsymbol{x}_i)}{\text{rad}\,(\boldsymbol{x}_i)}$, $i = 1, \ldots, n$.

3.3.2 Affine arithmetic

By replacing all the occurrences of the variable x_i by the corresponding affine form $\hat{x}_i$ in an expression of f, and by performing the computations using Affine Arithmetic, we obtain a resulting affine form, denoted by

$$\hat{f}(T(\boldsymbol{x}, \epsilon)) = r_0 + \sum_{i=1}^{n} r_i \epsilon_i + \sum_{k=n+1}^{N} r_k \epsilon_k, \tag{12}$$

where ϵ_j is in $[-1, 1]$ for all $j \in \{1, \ldots, N\}$. Note that some error terms $r_k \epsilon_k$ are added for all $k \in \{n + 1, \ldots, N\}$, which come from non affine operations in f.

3.3.3 Linear underestimation of f over $\boldsymbol{x}$

Using Affine Arithmetic, (12) underestimates f over $\boldsymbol{x}$

$$f(x) = f(T(\boldsymbol{x}, \epsilon)) \geq \underline{\hat{f}}(T(\boldsymbol{x}, \epsilon)) = r_0 + \sum_{i=1}^{n} r_i \epsilon_i - \sum_{k=n+1}^{N} |r_k|, \tag{13}$$

because all error terms are taken into account using their worst value.

Remark 7 Equation (13) is a linear underestimation of f over $\boldsymbol{x}$ using the new variables ϵ_i.

3.3.4 Linear program to provide lower bounds

In order to compute a lower bound of f over the simplex S (and not only on the $\boldsymbol{x}=\square S$), we constrain the point x to be inside S by adding (1). In this case, we describe x by its affine form $T(\boldsymbol{x}, \epsilon)$ and thus, we obtain the following linear program

$$\begin{aligned} \min_{\substack{\epsilon\in[-1,1]^n \\ \lambda\in[0,1]^{n+1}}} \quad & \sum_{i=1}^{n} r_i \epsilon_i \\ \text{s.t.} \quad & T(\boldsymbol{x}, \epsilon) = \sum_{i=0}^{n} \lambda_i v_i \\ & \sum_{i=0}^{n} \lambda_i = 1. \end{aligned} \tag{14}$$

Denoting the exact solution of (14) by (ϵ^*, λ^*), we have that

$$f(x) = f(T(\boldsymbol{x}, \epsilon)) \geq \sum_{i=1}^{n} r_i \epsilon_i^* + r_0 - \sum_{k=n+1}^{N} |r_k|, \ \forall x \in S \tag{15}$$

and therefore, this is a lower bound of f over S.

Note that to solve the above linear program, we just need to evaluate f at each vertex of S and then take the minimum value of the linear underestimation (13). If $\text{rad}\,(\boldsymbol{x}_i) \neq 0$ (else $r_i = 0$), $\epsilon_i(v_j)$ is a transformation of component i of vertex v_j into variable ϵ

$$\epsilon_i(v_j) = \begin{cases} \dfrac{(v_j)_i - \text{mid}\,(\boldsymbol{x}_i)}{\text{rad}\,(\boldsymbol{x}_i)}, & \text{if rad}\,(\boldsymbol{x}_i) > 0 \\ 0, & \text{if rad}\,(\boldsymbol{x}_i) = 0. \end{cases} \quad \forall i \in \{1, \ldots, n\}, \text{ and } \forall v_j \in \mathcal{V}.$$

Then, the lower bound of (15) becomes

$$\min_{j\in\{0,\ldots,n\}} \sum_{i=1}^{n} r_i \epsilon_i(v_j) + r_0 - \sum_{k=n+1}^{N} |r_k|. \tag{16}$$

Therefore, instead of solving linear program (14), we can determine (16) and this yields directly the lower bound of f over S. The solution corresponds to the solution of linear program (14).

4 Monotonicity test

In this paper, we use a concise monotonicity test which excludes an interior partition set S if it does not contain a stationary point. To be more precise:

Proposition 2 *Let $S \subset int(D)$ be a simplex in the interior of the search area D. If $\exists i \in \{1, \ldots, n\}$ with $0 \notin \nabla \boldsymbol{f}_i(\Box S)$ then S does not contain a global minimum point.*

Proof The condition implies that

$$\forall x \in S, \frac{\partial f}{\partial x_i}(x) \neq 0.$$

such that S cannot contain an interior minimum of D. Moreover, S does not touch the boundary, i.e. $S \cap \partial D = \emptyset$, such that neither it can contain a boundary optimum point. □

The test is not very strong, as initial partition sets typically touch the boundary. A slight relaxation is the following corollary, where a simplicial partition set has only vertices in common with the boundary.

Corollary 1 *Let $S \subset D$ be a partition set, where the number of boundary points is finite, i.e. $S \cap \partial D \subset \mathcal{V}$, and $0 \notin \nabla \boldsymbol{f}(\Box S)$. Then S can be eliminated from the search tree.*

Proof The same reasoning as in the proof of Proposition 2 applies with respect to the interior of S. Now a minimum point could be attained in a vertex $v \in S \cap \partial D$. However, vertex v is also part of another simplicial partition set which covers part of the boundary of D, such that we do not have to store S anymore. □

This corollary is not very strong, but it is relatively easy to check. For practical tests, Proposition 2 offers the conditions for removing an interior simplicial partition set S. We can also remove it, if just several vertices of S touch the boundary of D according to Corollary 1. Otherwise, we should store the facets of S which are completely in a face of the search region as new simplicial partition sets with less than $n + 1$ vertices.

In our former investigation [6], we focused on bounds of the directional derivative in a direction d, denoted by $\underline{d^T \nabla \boldsymbol{f}(S)}$ and $\overline{d^T \nabla \boldsymbol{f}(S)}$. In this context, one can consider for instance an upper bound of the directional derivative

$$\overline{d^T \nabla \boldsymbol{f}(S)} = \sum_{i=1}^{n} \max\{d_i \underline{\nabla \boldsymbol{f}}_i(\Box S), d_i \overline{\nabla \boldsymbol{f}}_i(\Box S)\}. \tag{17}$$

Notice that a necessary condition for $\overline{d^T \nabla \boldsymbol{f}(S)} \leq 0$ according to (17) is that f is monotone on $\Box S$, i.e. $0 \notin \nabla \boldsymbol{f}(\Box S)$.

Proposition 3 *Let $\mathcal{V}$ be a vertex set, $S = \mathrm{conv}(\mathcal{V})$, $w \in \mathcal{V}$, $\hat{V} = \mathcal{V} \setminus \{w\}$, facet $F = \mathrm{conv}(\hat{V})$ and $d = \frac{1}{n} \sum_{v \in \hat{V}} v - w$. If $\overline{d^T \nabla \boldsymbol{f}(S)} \leq 0$, then F contains a minimum point of $\min_{x \in S} f(x)$.*

Proof Consider the vertices of $\mathcal{V}$ ordered such that $w = v_0$. Let $x = \sum_{j=1}^{n} \lambda_j v_j + \lambda_0 w$ be a minimum point $x \notin F$. We construct a point z on F walking in direction d according to $z = x + \lambda_0 d = \sum_{j=1}^{n} (\lambda_j + \frac{1}{n}) v_j$. Then we have that $f(z) \leq f(x) + \lambda_0 \overline{\nabla d^T f^T(S)} \leq f(x)$. Thus, minimum point x either does not exist, or z is also a minimum point of $\min_{x \in S} f(x)$ and it is located on facet F. □

Corollary 2 *Let $\mathcal{V}$ be a vertex set, $S = \text{conv}(\mathcal{V})$, $w \in \mathcal{V}$, $\hat{V} = \mathcal{V} \setminus \{w\}$, facet $F = \text{conv}(\hat{V})$ and $d = \frac{1}{n}\sum_{v \in \hat{V}} v - w$. If $\overline{d^T \nabla \boldsymbol{f}(S)} < 0$, then F contains all minimum points of $\min_{x \in S} f(x)$, i.e $\text{argmin}_{x \in S} f(x) \subset F$.*

The corresponding test allows us to perform a dimension reduction of S by removing the vertex w. In case the conditions are not true, one can check each border facet if it can contain a minimum point. If we show it cannot, we do not have to deal further with the facet. In case no border facet can contain the minimum, it follows that S can be disregarded.

Corollary 3 *Let $\mathcal{V}$ be a vertex set, $S = \text{conv}(\mathcal{V})$, $w \in \mathcal{V}$, $\hat{V} = \mathcal{V} \setminus \{w\}$, facet $F = \text{conv}(\hat{V})$ and $d = \frac{1}{n}\sum_{v \in \hat{V}} v - w$. If $\underline{d^T \nabla \boldsymbol{f}(S)} > 0$, then F cannot contain a minimum point of $\min_{x \in S} f(x)$.*

5 Simplicial B&B algorithm (SBB)

Algorithm 1 uses an AVL tree[3] [1] Λ, a self-balancing binary search tree, for storing partition sets. Such a structure has a computational complexity of sorted insertion and extraction of an element of $\mathcal{O}(\log_2 |\Lambda|)$. Evaluated and not rejected simplices are sorted in Λ by non decreasing order of the bounds on the objective using any of the methods from Sect. 3. This means $[\underline{x}, \overline{x}] < [\underline{y}, \overline{y}]$ when $\underline{x} < \underline{y}$ or when $\underline{x} = \underline{y}$ and $\overline{x} < \overline{y}$. Simplicial partition sets having the same bounds are stored in the same node of the AVL tree using a linked list.

All vertices of a simplex are also stored in an AVL tree. Vertices may be shared among several simplices, such that we avoid duplicate storage. Although Algorithm 1 describes vertices to be evaluated in order to update the incumbent $\tilde{f}$ (see Algorithm 1, lines 5 and 13), their evaluation depends on the actual lower bounding method. The simplex S with the lowest value of $\boldsymbol{f}(S)$ is extracted from Λ (lines 8 and 19). The lower bound of $\boldsymbol{f}(S)$ is used in the stopping criterion of the algorithm (line 9).

Evaluation of a simplex S always includes computation of the natural inclusion $\boldsymbol{f}(S) = \boldsymbol{f}(\square S)$ of the objective function and inclusion of the gradient $\nabla \boldsymbol{f}(S) = \nabla \boldsymbol{f}(\square S)$ using Automatic Differentiation (see Algorithm 1, lines 4 and 16, and Algorithm 2 line 6). Other bounding methods can be applied afterwards in order to improve the calculated bounds in $\boldsymbol{f}$.

Simplices with a lower bound greater than the incumbent $\tilde{f}$ are rejected. They are also rejected using Proposition 2 when they are in the relative interior of the search space D and $0 \notin \nabla \boldsymbol{f}_i(\square S)$ (see Algorithm 1, lines 14 and 17, and Algorithm 2, line 7).

In case f is monotone on $\square S$ and $S \cap \partial D \neq \emptyset$, S can be reduced to a number of facets by Corollary 3 (see calls to Algorithm 2 from Algorithm 1, lines 6 and 10). From computational perspective it is better to label the vertex *border* or *not-border*. A *border* vertex means that when it is removed from S, the remaining facet is on the boundary of D. If the search region is a simplex, P contains just the initial simplex, and all initial facets are at the boundary, such that all vertices are labelled *border*. In case the search region is a box, P contains the result of the combinatorial vertex triangulation of the box into $n!$ simplices [24,26].

This technical detail has not been included in Algorithm 1 for the sake of simplicity. The specific triangulation is not appealing for large values of n. We use this here because box constrained problems are used to compare methods. Each of the $n!$ initial simplices has two border facets. They are determined by removing the smallest and largest vertex (numbered in a binary system), see the grey nodes in Fig. 1. In the binary system, 0 is the lower bound and 1 is the upper bound of the given component of the box.

[3] named after the inventors Adelson-Velsky and Landis

Algorithm 1 SBB(f, P, α)

Require:
f: the n dimensional objective function.
P: initial simplicial partition of the search region D.
α: termination criterion.
1: $\Lambda = \emptyset$ ▷ Storage structure
2: $\tilde{f} = \infty$ ▷ Incumbent value
3: **for** $S \in P$ **do**
4: Evaluate $\boldsymbol{f}(S)$, $\nabla \boldsymbol{f}(S)$ ▷ + other lower bounds
5: $\tilde{f} \leftarrow \min\{\tilde{f},\ \min_{v_j \in S} \overline{\boldsymbol{f}}(v_j)\}$
6: **if** $\underline{\boldsymbol{f}}(S) \le \tilde{f}$ **and not** ReduceToFacets($\boldsymbol{f}$,S,Λ) **then**
7: $\Lambda \leftarrow S$ ▷ Store S and its bounds in Λ
8: $S \leftarrow \Lambda$ ▷ Retrieve S from Λ with smallest $\underline{\boldsymbol{f}}(S)$ value
9: **while** wid ($[\underline{\boldsymbol{f}}(S)), \tilde{f}]) > \alpha$ **do**
10: **if not** ReduceToFacets($\boldsymbol{f}$,S,Λ) **then**
11: $\{S_1, S_2\} \leftarrow$ Divide(S) ▷ Longest Edge Bisection
12: **if** $\overline{\boldsymbol{f}}$(new vertex) $< \tilde{f}$ **then**
13: $\tilde{f} = \overline{\boldsymbol{f}}$(new vertex)
14: $\Lambda =$ CutOff(Λ) ▷ Remove $S \in \Lambda : \underline{\boldsymbol{f}}(S) > \tilde{f}$
15: **for** each subset S_j **do**
16: Evaluate $\boldsymbol{f}(S_j)$, $\nabla \boldsymbol{f}(S_j)$ ▷ + other lower bounds
17: **if** $\underline{\boldsymbol{f}}(S_j) \le \tilde{f}$ **and not** (Monotone **and** $\nexists$ vertex labelled *border*) **then** ▷ See Proposition 2
18: $\Lambda \leftarrow S_j$ ▷ Store S_j and its bounds in Λ
19: $S \leftarrow \Lambda$
20: **return** $[\underline{\boldsymbol{f}}(S), \tilde{f}]$

A simplicial partition set, which was neither rejected nor reduced, is divided using Longest Edge Bisection (LEB), see Algorithm 1, line 11. When several longest edges exist, the longest edge with a vertex with the lowest value of $\underline{\boldsymbol{f}}$ and the other vertex having the highest value of $\underline{\boldsymbol{f}}$ is selected. In case vertices are not evaluated, the first longest edge is selected.

Remark 8 The interior of a new facet generated by LEB is always in the relative interior of the bisected simplex. This contributes to reduce the number of vertices labelled as *border* in the new sub-simplices.

Descendants of a partition set having all its vertices labelled as *not-border* have all facets in the interior of D, so labelling is no longer necessary.

Algorithm 2 ReduceToFacets($\boldsymbol{f}$,S,Λ)

Reduced=**false**
if Mon($\boldsymbol{f}$,S) **and** $S \cap \partial D \neq \emptyset$ **then** ▷ See Proposition 2
for each border facet F **do**
if not $d^T \nabla \boldsymbol{f}(S) > 0$ **then** ▷ See Corollary 3
Reduced=**true**
Evaluate $\boldsymbol{f}(F)$, $\nabla \boldsymbol{f}(F)$ ▷ + other lower bounds
if $\underline{\boldsymbol{f}}(F) \le \tilde{f}$ **and not** (Monotone **and** $\nexists$ vertex labelled *border*) **then** ▷ See Proposition 2
$\Lambda \leftarrow F$ ▷ Store F and its bounds in Λ
return Reduced

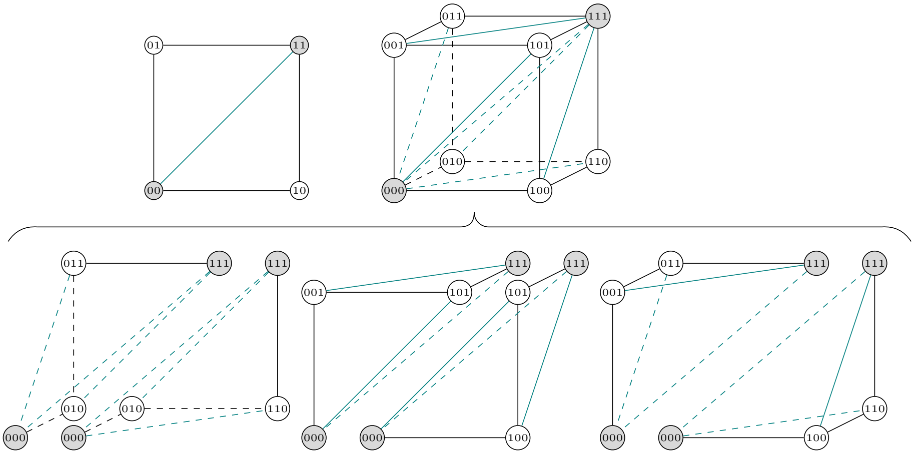

Fig. 1 Combinatorial vertex triangulation of an hyper-rectangle. Vertices 000 and 111 are labelled *border* in all sub-simplices. Removing one of them leaves a facet that is completely on the boundary of D

6 Numerical results

Algorithm 1 was run on an Asus UX301L NoteBook with Intel(R) Core(TM) i7-4558U CPU and 8GB of RAM running Fedora 32 Linux distribution. The algorithm was coded with g++ (gcc version 10.1.1) and it uses Kv-0.4.50 for Interval Arithmetic and Affine Arithmetic (AA). Kv uses boost libraries. Algorithms were compiled with `-O3 -DNDEBUG -DKV_FASTROUND` options and AA uses `#define AFFINE_SIMPLE 2` and `#define AFFINE_MULT 2` in Kv. For the Linear Programming, we use PNL 1.10.4 with `-DCMAKE_BUILD_TYPE = Release -DWITH_MPI = OFF`, as a C++ wrapper to LPsolve 5.5.2.0.

Notice that Kv Affine Arithmetic is slow in execution speed: *When the direction of rounding is fixed as "upward", the downward calculation is performed as "sign inversion"*, and it currently does not support division by affine variables containing zero. Additionally, the execution time for Interval Arithmetic can be reduced on processors supporting Advanced Vector Extensions SIMD (AVX-512) (see last table at kv-rounding web page), which is not our case. Moreover, the PNL library has support for MPI, which is not used here.

Table 1 describes the studied instances. Their detailed description can be found in [7] and the optimization web page.

The used termination accuracy is $\alpha = 10^{-6}$ and Interval Arithmetic is applied with Automatic Differentiation to obtain bounds of f and ∇f on $\square S$. The following notation is used to describe the variants to calculate lower bounds:

- IA : Natural IA.
- +CFcb : IA + Centered form on a box (see Not. 8) using the center of $\square S$.
- +CFbb : IA + Centered form on a box (see Not. 8) using the Baumann point bb^- on $\square S$ (see Not. 9).
- +CFcs : IA + Centered form on a simplex (see Prop. 1) using the centroid as the base-point and the gradient on $\square S$.
- +CFbs : IA + Centered form on a simplex (see Prop. 1) using base-point $y = bs^-$ (see Def. 1) and the gradient on $\square S$.

Table 1 Test problems. The problems are box constrained apart from KE2-1 and KE2-2 with search regions {(−3,-1),(1,1),(1.5,−2)} and {(−2,0),(0,−3),(2,3)}, respectively. An asterisk at n indicates that this is the selected dimension for a varying dimension test instance

Instance	Description	n
KE2-1	Karhbet example 6 over simplex 1	2
KE2-2	Karhbet example 6 over simplex 2	2
GP2	Goldstein-Price	2
THCB2	Three Hump Camel Back	2
SHCB2	Six Hump Camel Back	2
G7	Griewank	7*
S4	Shekel 10	4
H3	Hartmann 3	3
H4	Hartmann 4	4
H6	Hartmann 6	6
L8	Levy	8*
SCH2	Schubert	2
MC2	McCormick	2
RB2	Rosenbrock	2*
MCH2	Michalewicz	2*
MCH5	Michalewicz	5*
ST2	Styblinski-Tang	2*
ST5	Styblinski-Tang	5*
DP2	Dixon-Price	2*
DP5	Dixon-Price	5*

+CFvs : IA + Centered form on a simplex (see Prop. 1) using base-point $y = \underset{v \in S}{\operatorname{argmax}}\{\overline{f}(v)\}$ and the gradient on $\square S$.
+AA : IA + Affine Arithmetic lower bound (16) over $\square S$.
+LR : IA + Linear Relaxation bound (9) on $\square S$.
+LRS : IA + Linear Relaxation bound (10) on $\square S$, forcing the solution on S.

Rejection tests like the ones on monotonicity, are checked after the bound calculations. This is not efficient, but it allows us to compare the calculated bounds.

Improvement of the best function value found $\tilde{f}$ is done by point evaluation. Together with the IA bound calculation we evaluate simplex vertices. When other lower bound methods are added to IA, the evaluation of simplex vertices can be disabled in order to save computation. However, this may imply another (worse) update of $\tilde{f}$ and a different course of the algorithm, due to Longest Edge Bisection (LEB) by the first longest edge, instead of the best LEB [19].

The following points are evaluated for each method. +CFc* methods (*=b or s) evaluate only the center and +CFb* evaluate only base-points bb^- or bs^-. Such points are not stored. Notice that base point bb^- might be located outside the simplicial search region. +CFvs and +AA evaluate and store simplex vertices. +LR and +LRS evaluate and store box vertices when the search region is a box. Additionally, simplex vertices are evaluated when the search region is a simplex, because vertices of $\square S$ may be outside the search region and should not be used to improve $\tilde{f}$.

The +CF*s (*=c,b or v) methods only update lower bounds. The other methods also update upper bounds, which may affect the partition set storage order.

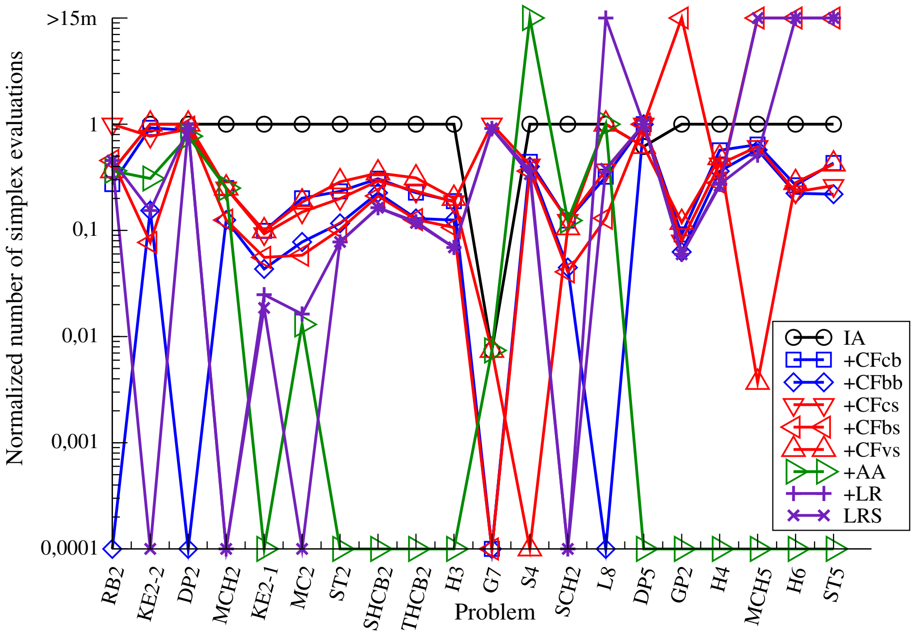

Fig. 2 Normalized number of simplex evaluations in log scale. A value of $> 15m$ means time out of 15 minutes or execution error. The ranges of evaluated simplices per problem are as follow: RB2$\in$ [44, 66], KE2-2$\in$ [47, 60], DP2$\in$ [16, 112], MCH2$\in$ [128, 192], EX2-1$\in$ [186, 510], MC2$\in$ [434, 1,052], ST2$\in$ [558, 1,382], SHCB$\in$ [556, 1,646], THCB$\in$ [626, 1,986], H3$\in$ [2,286, 4,430], G7$\in$ [5,040, 5,314], S4 $\in$ [3,984, 5,288], SCH2$\in$ [4,862, 6,834], L8$\in$ [40,462, 40,662], DP5$\in$ [97,060, 188,476], GP2$\in$ [2,272, 167,800], H4$\in$ [53,368, 170,622], MCH5$\in$ [189,198, 210,356], H6$\in$ [2,641,024, 4,944,040], and ST5$\in$ [2,569,082, 6,358,328]

Figures 2 and 3 show the normalized (to the range [0,1]) number of simplex evaluations (NS) and execution time (T), respectively. The number of simplex evaluations can be considered as the number of iterations, as in each iteration one simplex is evaluated. The problems are sorted by NS in both figures. The data for the figures is taken from Table 2 to Table 21 in Appendix A. Reduction to border facets due to monotonicity does not occur in box constrained problems. It happens in the simplex constrained instances (see Corollary 3). The monotonocity test reduces the number of simplex evaluations significantly for all test problems. Without that test, the algorithm lasts more than the limit of 15 minutes for several problems. Therefore, we always apply the monotonicity test.

Going over the results of the test problems, problem G7 appears to be a special case, see Table 7. Apparently, adding methods to IA does not provide better lower bounds. For L8 only the +AA method improves the bound a few times and for RB2 only the LR* methods show tighter bounds, see Table 12.

A value of $> 15m$ in Figs. 2 and 3 means that i) the algorithm reached the 15 minute time limit, or ii) there was a problem with the Linear Programming solver in methods +LR and +LRS or iii) a division by zero occurred. The latter only happens for the +AA method for problem L8, because the Kv library does not implement division by zero in Affine Arithmetic.

Focusing on the number of required simplex evaluations (NS), Fig. 2 shows that the +AA method requires the least evaluations for most of the test cases. The second best methods regarding the NS metric are those using LP (+LR* and +CFbs). The +LRS lower bounding

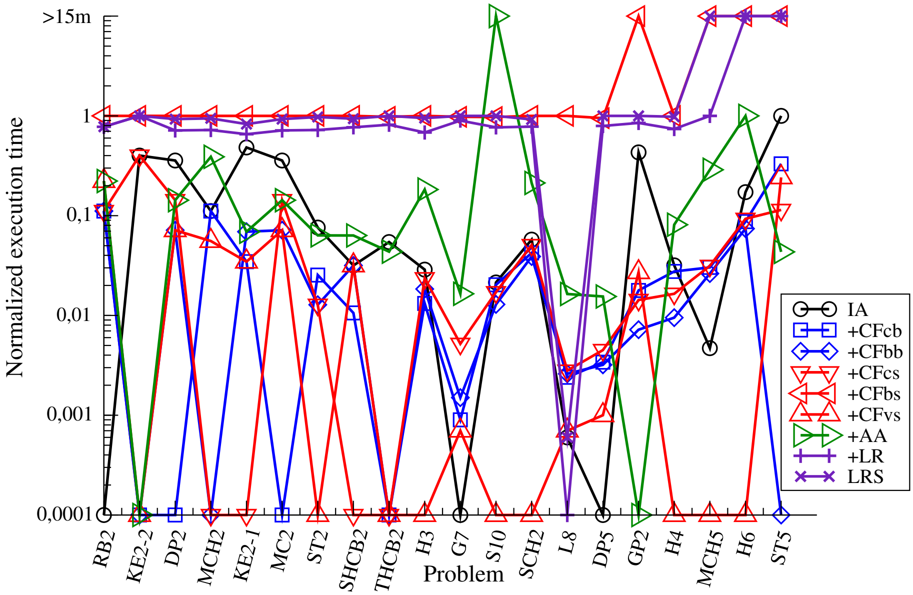

Fig. 3 Normalized execution time (seconds) in log scale. A value of $> 15m$ means time out of 15 minutes or execution error. The ranges of execution time per problem are as follow: RB2∈ [0.005, 0.014], EX2-2∈ [0.005, 0.010], DP2∈ [0.005, 0.019], MCH2∈ [0.006, 0.024], EX2-1∈ [0.005, 0.034], MC2∈ [0.005, 0.019], ST2∈ [0.006, 0.085], SHCB2∈ [0.006, 0.101], THCB2∈ [0.008, 0.1], H3∈ [0.032, 0.415], G7∈ [0.048, 6.69], S4 ∈ [0.056, 0.995], SCH2∈ [0.104 0.696], L8∈ [0.566, 159.124], DP5∈ [0.526, 61.352], GP2∈ [0.051, 1.461], H4∈ [1.196, 19.844], MCH5∈ [3.611, 60.438], H6∈ [51.057, 187.538], and ST5∈ [58.476, 184.924]

Table 2 Results for KearEx6 on {(−3,−1),(1,1),(1.5,−2)} simplex. Global min. is interior. +CFcb and +CFcs show vertex evaluations due to reduction

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	510	165	12	165	0	0	0	0.019s
+CFcb	218	4	10	73	218	0	213	0.006s
+CFbb	200	67	9	67	200	0	199	0.007s
+CFcs	216	4	10	73	216	0	213	**0.005s**
+CFbs	204	0	8	69	204	0	204	0.034s
+CFvs	218	74	9	74	0	0	212	0.006s
+AA	**186**	63	8	63	0	0	186	0.007s
+LR	194	66	9	397	0	331	194	0.024s
+LRS	192	66	8	391	0	325	192	0.029s

requires less simplex evaluations than +LR for some cases, and +CFcb has the best NS values for only a few test cases.

For smooth functions, the algorithm converges to a region which is captured by a convex quadratic function. To study the limit convergence behaviour of the algorithm, we run all variants over the so-called Trid function from [23], which represents a convex quadratic function. The results can be found in Tables 22–24 (Appendix B). One can observe for this limit situation that the Linear Relaxation variants are relatively close models and require less simplex evaluations than other lower bounding methods. This means that, for all cases, the

Table 3 Results for KearEx6 on {(−2,0),(0,−3),(2,3)} simplex. Global Min at border.+CFcb and +CFcs show vertex evaluations due to reduction

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	60	27	7	27	0	0	0	0.007s
+CFcb	59	8	5	27	59	0	51	**0.005s**
+CFbb	49	24	5	24	49	0	48	**0.005s**
+CFcs	57	8	5	26	57	0	53	0.007s
+CFbs	48	1	4	24	48	0	47	0.010s
+CFvs	60	27	6	27	0	0	52	**0.005s**
+AA	51	24	4	24	0	0	49	**0.005s**
+LR	49	24	4	86	0	62	48	0.010s
+LRS	**47**	23	3	81	0	58	46	0.010s

Table 4 Results for Goldstein-Price on box $[-2, 2]^2$. (1) Unacceptable accuracy found (worse than required 5e-07). linprog fails

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	167,800	43,288	10,924	43,288	0	0	0	0.658s
+CFcb	17,442	0	1,407	4,589	17,442	0	15,838	0.076s
+CFbb	12,640	0	1,036	3,371	12,640	0	11,737	0.061s
+CFcs	15,352	0	1,153	4,046	15,352	0	13,995	0.071s
+CFbs	(1)							
+CFvs	21,910	5,936	1,701	5,936	0	0	19,570	0.089s
+AA	**2,272**	683	177	683	0	0	2,165	**0.051s**
+LR	12,118	0	959	4,788	0	4,282	11,253	1.249s
+LRS	12,080	0	950	4,788	0	4,282	11,229	1.461s

Table 5 Results for Three Hump Camel Back on box $[-5, 5]^2$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	1,986	581	132	581	0	0	0	0.013s
+CFcb	934	0	58	281	934	0	508	**0.008s**
+CFbb	802	0	52	245	802	0	558	**0.008s**
+CFcs	944	0	58	290	944	0	556	**0.008s**
+CFbs	798	0	50	245	798	0	564	0.098s
+CFvs	1,050	317	58	317	0	0	516	**0.008s**
+AA	**626**	183	32	183	0	0	494	0.012s
+LR	790	0	46	375	0	373	632	0.083s
+LRS	786	0	48	375	0	373	628	0.100s

+LR variants have an advantage in the final stages of the algorithm. It is worth to mention that, when the dimension increases, the required Linear Programming gets more time consuming and also the +AA variant starts to do better.

The execution time is a difficult performance indicator, as it depends on the used external subroutines. Figure 3 provides normalized values. In the first 9 test cases (ordered according to NS), the execution time is similar for most of the methods apart from those using LP

Table 6 Results for Six Hump Camel Back on box $[-3, 3] \times [-2, 2]$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	1,646	535	108	535	0	0	0	0.009s
+CFcb	890	0	78	291	890	0	642	0.007s
+CFbb	802	0	64	267	802	0	664	0.009s
+CFcs	874	0	66	287	874	0	668	**0.006s**
+CFbs	786	0	66	261	786	0	656	0.101s
+CFvs	934	305	64	305	0	0	618	0.009s
+AA	**556**	198	41	198	0	0	522	0.012s
+LR	734	0	70	385	0	377	610	0.079s
+LRS	734	0	70	385	0	377	610	0.095s

Table 7 Results for Griewank on box $[-600, 600]^7$. Notice that 7!=5,040

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	5,042	129	5,041	129	0	0	0	**0.048s**
+CFcb	**5,040**	0	5,040	128	5,040	0	0	0.053s
+CFbb	**5,040**	5,040	128	5,040	0	0	0	0.057s
+CFcs	5,314	0	5,088	265	5,314	0	0	0.082s
+CFbs	**5,040**	0	5,040	128	5,040	0	0	6.551s
+CFvs	5,042	129	5,041	129	0	0	0	0.052s
+AA	5,042	129	5,041	129	0	0	0	0.157s
+LR	5,290	0	5,078	13,165	0	13,040	0	6.005s
+LRS	5,290	0	5,078	13,165	0	13,040	0	6.690s

Table 8 Results for Shekel 10 on box $[0, 10]^4$. (1):affine: division by 0

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	5,288	470	345	470	0	0	0	0.076s
+CFcb	4,560	0	301	378	4,560	0	1,172	0.075s
+CFbb	4,504	0	76	367	4,504	0	1,244	0.068s
+CFcs	4,520	0	313	373	4,520	0	1,232	0.072s
+CFbs	4,456	0	76	361	4,456	0	1,300	0.970s
+CFvs	**3,984**	355	345	355	0	0	1,164	**0.056s**
+AA	(1)							
+LR	4,466	0	383	9455	0	9323	1962	0.776s
+LRS	4,424	0	340	5,333	0	5,277	1,508	0.995s

(+LR* and +CFbs). In fact, methods using LP are in general the most time consuming due to the called routines, followed by +AA which avoids solving an LP due to (16). According to the Kv library documentation, Affine Arithmetic is slow and its implementation could be improved.

Table 9 Results for Hartmann3 on box $[0, 1]^3$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	4,430	617	116	617	0	0	0	0.043s
+CFcb	2,690	0	100	378	2,690	0	1,815	0.037s
+CFbb	2,554	0	95	364	2,554	0	1,971	0.039s
+CFcs	2,684	0	98	378	2,684	0	1,895	0.041s
+CFbs	2,514	0	95	362	2,514	0	1,980	0.415s
+CFvs	2,714	383	116	383	0	0	1,702	**0.032s**
+AA	**2,286**	336	94	336	0	0	1,919	0.102s
+LR	2,434	0	95	880	0	870	1,851	0.293s
+LRS	2,434	0	95	880	0	870	1,856	0.396s

Table 10 Results for Hartmann4 on box $[0, 1]^4$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	170,622	11,125	7,686	11,125	0	0	0	1.788s
+CFcb	119,942	0	7,022	9,211	119,942	0	69,387	1.709s
+CFbb	95,636	0	5,497	7,310	95,636	0	70,851	1.371s
+CFcs	102,678	0	5,393	7,745	102,678	0	69,827	1.501s
+CFbs	86,138	0	4,483	6,391	86,138	0	66,847	19.844s
+CFvs	110,130	7,002	6,535	7,002	0	0	64,093	**1.196s**
+AA	**53,368**	3,545	3,416	3,545	0	0	44,630	2.709
+LR	85,152	0	4,935	160,048	0	157,273	67,065	14.965s
+LRS	84,092	0	4,857	158,012	0	155,251	66,557	19.472s

Table 11 Results for Hartmann6 on box $[0, 1]^6$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	4,944,040	51,671	112,878	51,671	0	0	0	1m14.548s
+CFcb	3,253,420	0	105,566	39,135	3,253,420	0	1,495,023	1m2.928s
+CFbb	3,155,816	0	105,421	37,652	3,155,816	0	1,739,241	1m1.153s
+CFcs	3,170,950	0	105,794	37,882	3,170,950	0	1,635,979	1m3.736s
+CFbs								> 15m.
+CFvs	3,288,994	32,677	112,890	32,677	0	0	1,742,240	**51.057s**
+AA	**2,641,024**	26,308	106,304	26,308	0	0	1,690,829	3m7.538s
+LR								> 15m.
+LRS								> 15m.

The +CFvs method requires the least execution time for most of the instances. Comparing +CFvs with other +CF* methods, the centered form used in +CFvs has to evaluate one sum term less and the base-point vertex can already have been evaluated and stored. On average, +CFbb is the best method, but this is because it is the best for the ST5 test problem, which is one of the most time consuming instances.

Table 12 Results for Levy 8 on $[-10, 10]^8$ box. (1)Unacceptable accuracy found (worse than required 5e-07). linprog fails

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	40,662	381	40,353	381	0	0	0	**0.640s**
+CFcb	40,526	0	40,352	359	40,526	0	0	0.924s
+CFbb	**40,462**	0	40,351	327	40,462	0	0	0.958s
+CFcs	40,536	0	40,352	364	40,536	0	0	0.990s
+CFbs	40,488	0	40352	340	40488	0	0	2m39.124s
+CFvs	40,662	381	40,353	381	0	0	0	0.666s
+AA	40,662	381	40,353	381	0	0	3	3.150s
+LR	(1)							
+LRS	40,538	0	40,355	27,037	0	26,928	0	2m34.620s

Table 13 Results for Schubert on box $[-10, 10]^2$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	6,834	2,297	1,076	2,297	0	0	0	0.138s
+CFcb	5,106	0	655	1,649	5,106	0	1,742	0.129s
+CFbb	4,950	0	639	1,595	4,950	0	1,756	0.127s
+CFcs	5,106	0	599	1,649	5,106	0	1,806	0.134s
+CFbs	4,942	0	655	1,597	4,942	0	1,770	0.696s
+CFvs	5,070	1,649	1,076	1,649	0	0	1,726	**0.104s**
+AA	5,106	1,649	1,076	1,649	0	0	1,788	0.230s
+LR	**4,862**	0	1,076	2,553	0	2,450	1,752	0.566s
+LRS	**4,862**	0	1,076	2,553	0	2,450	1,758	0.649s

Table 14 Results for McCormick on box $[-1.5, 4] \times [-3, 4]$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	1,052	329	24	329	0	0	0	0.010s
+CFcb	558	0	22	178	558	0	487	0.009s
+CFbb	482	0	20	149	482	0	451	0.007s
+CFcs	526	0	19	165	526	0	473	0.010s
+CFbs	470	0	19	145	470	0	442	0.065s
+CFvs	548	177	21	177	0	0	477	0.009s
+AA	442	141	18	141	0	0	427	0.009s
+LR	444	0	21	239	0	228	424	0.048s
+LRS	**434**	0	19	239	0	228	414	**0.062s**

Table 15 Results for Rosenbrock on box $[-5, 10]^2$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	52	29	16	29	0	0	0	**0.005s**
+CFcb	**50**	0	15	28	50	0	0	0.006s
+CFbb	44	0	12	25	44	0	0	0.006s
+CFcs	66	0	21	36	66	0	0	0.006s
+CFbs	54	0	16	30	54	0	0	0.014s
+CFvs	52	29	16	29	0	0	0	0.007s
+AA	52	29	16	29	0	0	0	0.007s
+LR	54	0	17	69	0	69	3	0.012s
+LRS	54	0	17	69	0	69	3	0.012s

Table 16 Results for Michalewicz on box $[0, 3.1416]^2$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	192	74	9	74	0	0	0	0.008s
+CFcb	144	0	9	56	144	0	66	0.008s
+CFbb	136	0	4	53	136	0	78	**0.006s**
+CFcs	144	0	12	56	144	0	67	**0.006s**
+CFbs	136	0	6	53	136	0	75	0.024s
+CFvs	144	56	9	56	0	0	63	0.007s
+AA	144	56	9	56	0	0	66	0.013s
+LR	**128**	0	9	81	0	81	70	0.019s
+LRS	**128**	0	9	81	0	81	70	0.023s

Table 17 Results for Michalewicz on box $[0, 3.1416]^5$. (1) matrix contains zero-valued coefficients. (2) Unacceptable accuracy found (worse than required 5e-07). linprog fails

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	210,356	6,964	14,087	6,964	0	0	0	3.870s
+CFcb	202,790	0	12,031	7,236	202,790	0	25,216	5.329s
+CFbb	201,320	0	8,935	7,172	201,320	0	30,857	5.098s
+CFcs	202,200	0	9,304	7,203	202,200	0	27,757	5.389s
+CFbs	(1,2)							
+CFvs	189,274	6,153	14,087	6,153	0	0	26,046	**3.611s**
+AA	**189,198**	6,150	14,087	6,150	0	0	26,217	19.981s
+LR(1)	200,016	0	12,791	1,001,262	0	996,620	34,764	1m0.438
+LRS	(1,2)							

7 Conclusions

In simplicial branch and bound methods, the determination of the lower bound is of great importance. The Interval Arithmetic lower bound on a simplex interval hull can be tightened by additional calculations at a given cost. Several methods have been described and investigated. We have used the centered form with several base-points over a simplex and the interval hull of a simplex. The use of Affine Arithmetic and a Linear Relaxation over the

Table 18 Results for Styblinski-Tang on $[-5, 5]^2$ box

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	1,382	444	62	444	0	0	0	0.012s
+CFcb	750	0	54	246	750	0	642	0.008s
+CFbb	654	0	42	216	654	0	608	0.007s
+CFcs	722	0	48	242	722	0	632	0.007s
+CFbs	642	0	38	214	642	0	604	0.085s
+CFvs	794	263	56	263	0	0	608	**0.006s**
+AA	**558**	182	36	182	0	0	552	0.011s
+LR	622	0	41	305	0	292	584	0.063s
+LRS	622	0	41	305	0	292	584	0.083s

interval hull and over the simplex has also been presented. Moreover, we introduced several theoretical results about monotonicity that can be applied to construct new rejection tests.

Results on a set of well known low dimensional test instances show that Affine Arithmetic is a promising method to get lower bounds over a simplex. It requires the smallest number of simplex evaluations in many problems. However, its computational time is larger than that of several other methods. In general, methods using a Linear Programming solver suffer the same drawback requiring more time. We found that the monotonicity tests were essential for the reduction of computing time.

Table 19 Results for Styblinski-Tang on box $[-5, 5]^5$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	36,358,328	843,581	1,738,800	843,581	0	0	0	3m04.924s
+CFcb	17,136,600	0	1,472,281	481,742	17,136,600	0	14,866,440	1m40.292s
+CFbb	9,976,680	0	1,153,440	223,784	9,976,680	0	9,727,920	**58.476s**
+CFcs	11,479,080	0	1,033,320	293,839	11,479,080	0	10,616,760	1m12.900s
+CFbs								> 15m
+CFvs	16,777,800	341,005	1,309,801	341,005	0	0	13,589,880	1m29.063s
+AA	**2,569,082**	43,398	266,040	43,398	0	0	2,568,122	1m3.954s
+LR								> 15m
+LRS								> 15m

Table 20 Results for Dixon-Price on box $[-10, 10]^2$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	112	50	18	50	0	0	0	0.010s
+CFcb	100	0	16	44	100	0	0	**0.005s**
+CFbb	**16**	0	5	11	16	0	2	0.006s
+CFcs	102	0	14	45	102	0	0	0.007s
+CFbs	106	0	15	47	106	0	11	0.019s
+CFvs	112	50	18	50	0	0	0	0.006s
+AA	90	40	18	40	0	0	10	0.007s
+LR	104	0	18	73	0	71	12	0.015s
+LRS	104	0	18	73	0	71	13	0.018s

Table 21 Results for Dixon-Price on box $[-10, 10]^5$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	153,392	4,250	6,395	4,250	0	0	0	**0.526s**
+CFcb	188,476	0	7,190	5,305	188,476	0	0	0.725s
+CFbb	188,362	0	2,314	5,279	188,362	0	360	0.714s
+CFcs	188,420	0	7,257	5,300	188,420	0	17	0.794s
+CFbs	188,376	0	7,180	5,286	188,376	0	1,664	58.210s
+CFvs	153,392	4,250	6,395	4,250	0	0	12	0.578s
+AA	**97,060**	2,917	5,676	2,917	0	0	22,387	1.461s
+LR	188,384	0	2,648	474,561	0	471,384	799	48.791s
+LRS	188,384	0	2,648	47,4561	0	471,384	8,226	1m1.352s

Table 22 Results for Trid on box $[-4, 4]^2$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	1,122	334	20	334	0	0	0	0.008s
+CFcb	578	0	20	174	578	0	532	**0.005s**
+CFbb	298	0	14	125	298	0	278	0.007s
+CFcs	570	0	20	170	570	0	534	0.006s
+CFbs	298	0	16	125	298	0	283	0.045s
+CFvs	574	175	18	175	0	0	528	0.006s
+AA	386	105	10	105	0	0	372	0.008s
+LR	**282**	0	18	129	0	128	266	0.030s
+LRS	**282**	0	18	129	0	128	268	0.043s

The method requiring least computing time in several test problems is the one based on the center form on a simplex using the vertex of a simplex with the highest function value as a base-point. The vertex can already have been evaluated and stored and the centered form requires one less additional term evaluation.

Table 23 Results for Trid on box $[-9, 9]^3$

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	9,862	1,421	114	1,421	0	0	0	0.024s
+CFcb	4,930	0	98	715	4,930	0	4,466	0.016s
+CFbb	4,106	0	96	597	4,106	0	3,916	**0.015s**
+CFcs	4,690	0	102	683	4,690	0	4,332	0.017s
+CFbs	3,860	0	86	556	3,860	0	3,738	0.571s
+CFvs	4,926	721	116	721	0	0	4,468	0.016s
+AA	3,526	533	84	533	0	0	3,428	0.033s
+LR	3,658	0	96	1,486	0	1,458	3,584	0.412s
+LRS	**3,510**	0	102	1,478	0	1,450	3,450	0.515s

This means that it is preferable to evaluate cheap lower bounds that reuse previous information over more simplices than expensive lower bounds over less simplices for low dimensional instances.

Acknowledgements This research is supported by the Spanish Ministry (RTI2018-095993-B-I00), in part financed by the European Regional Development Fund (ERDF).

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

A Extended numerical results

In Tables 2 to 21 the following notation is used.

NS : number of simplex evaluations,
NSV : number of simplex vertex evaluations,
MTS : maximum number of simplices stored in the AVL tree,
MTP : maximum number of points stored in the AVL tree,
NNV : number of non simplex vertex evaluations. They can be cb, bb^-, cs or bs^-.
NBV : number of $\Box S$ vertex evaluations,
NI : number of times the natural inclusion lower bound is improved by another lower bounding method,
T : wall clock time. Differences smaller than 0.005s are not significant.

Table 24 Results for Trid on box $[-25, 25]^5$. (1): Unacceptable accuracy found (worse than required 5e-07). linprog fails

LB	NS	NSV	MTS	MTP	NNV	NBV	NI	T
IA	10,891,264	289,293	46,300	289,293	0	0	0	42.946s
+CFcb	5,018,560	0	57,116	141,508	5,018,560	0	4,837,484	22.528s
+CFbb	3,929,044	0	64,524	106,795	3,929,044	0	3,878,012	**18.440s**
+CFcs	4,070,716	0	57,112	109,902	4,070,716	0	4,003,534	21.166s
+CFbs	(1)							
+CFvs	4,357,124	107,706	57,256	107,706	0	0	4,247,010	19.388s
+AA	**1,995,960**	47,597	38,994	47,597	0	0	1,991,164	28.440s
+LR	3,080,752	0	53,436	18,208,468	0	18,160,614	3,075,286	14m20.630s
+LRS	(1)							

B Methods on a convex quadratic function

The Trid problem from optimization is a convex quadratic function. According to Some Hard Global Optimization Test Problems: This is a simple discretized variational problem, convex, quadratic, with a unique local minimizer and a tridiagonal Hessian. The scaling behaviour for increasing n (search region is $[-n^2, n^2]$) gives an idea on the efficiency of localizing minima once the region of attraction (which here is everything) is found; most local methods only need $O(n^2)$ function evaluations, or only $O(n)$ (function+gradient) evaluations. A global optimization code that has difficulties with solving this problem for n=100, say, is of limited worth only. The strong coupling between the variables causes difficulties for genetic algorithms. The problem is typical for many problems from control theory, though the latter are usually nonquadratic and often nonconvex.

References

1. Adelson-Velsky, G.M., Landis, E.M.: An algorithm for the organization of information. Proceed. USSR Acad. Sci. (in Russian) **146**, 263–266 (1962)
2. Andrade, A., Comba, J., Stolfi, J.: Affine arithmetic. International Conf. on Interval and Computer-Algebraic Methods in Science and Engineering (INTERVAL/94) (1994)
3. Baumann, E.: Optimal centered forms. BIT Num. Math. **28**(1), 80–87 (1988). https://doi.org/10.1007/BF01934696
4. de Figueiredo, L., Stolfi, J.: Affine arithmetic: concepts and applications. Num. Alg. **37**(1–4), 147–158 (2004). https://doi.org/10.1023/B:NUMA.0000049462.70970.b6
5. Hansen, E., Walster, W.: Global Optimization Using Interval Analysis, 2ème edn. Marcel Dekker Inc., New York (2004)
6. Hendrix, E.M.T., Salmerón, J.M.G., Casado, L.G.: On function monotonicity in simplicial branch and bound. AIP Conf. Proceed. (2019). https://doi.org/10.1063/1.5089974
7. Karhbet, S.D., Kearfott, R.B.: Range bounds of functions over simplices, for branch and bound algorithms. Reliable Computing **25**, 53–73 (2017). https://interval.louisiana.edu/reliable-computing-journal/volume-25/reliable-computing-25-pp-053-073.pdf
8. Kearfott, R.B.: Rigourous Global Search: Continuous Problems. Kluwer Academic Publishers, Newyork (1996)
9. Messine, F.: Extensions of affine arithmetic: application to unconstrained global optimization. J. Univ. Comput. Sci. **8**(11), 992–1015 (2002). https://doi.org/10.3217/jucs-008-11-0992
10. Messine, F., Lagouanelle, J.L.: Enclosure methods for multivariate differentiable functions and application to global optimization. J. Univ. Comput. Sci. **4**(6), 589–603 (1998). https://doi.org/10.3217/jucs-004-06-0589
11. Messine, F., Touhami, A.: A general reliable quadratic form: an extension of affine arithmetic. Reliab. Comput. **12**(3), 171–192 (2006). https://doi.org/10.1007/s11155-006-7217-4
12. Mohand, O.: Tighter bound functions for nonconvex functions over simplexes. RAIRO Oper. Res. **55**, S2373–S2381 (2021). https://doi.org/10.1051/ro/2020088
13. Moore, R.: Interval Analysis. Prentice-Hall Inc., Englewood Cliffs (1966)
14. Ninin, J., Messine, F., Hansen, P.: A reliable affine relaxation method for global optimization. 4OR **13**(3), 247–277 (2014). https://doi.org/10.1007/s10288-014-0269-0
15. Paulavičius, R., Žilinskas, J.: Simplicial Global Optimization. Springer, New York (2014)
16. Rall, L.B.: Automatic differentiation: Techniques and applications. Lecture Notes in Computer Science, vol. 120. Springer, Newyork (1981)
17. Ratschek, H., Rokne, J.: Computer Methods for the Range of Functions. Ellis Horwood Ltd, Chichester (1984)
18. Rump, S.M., Kashiwagi, M.: Implementation and improvements of affine arithmetic. Nonlin. Theory Appl. **6**(3), 341–359 (2015). https://doi.org/10.1587/nolta.6.341
19. Salmerón, J.M.G., Aparicio, G., Casado, L.G., García, I., Hendrix, E.M.T., Toth, B.G.: Generating a smallest binary tree by proper selection of the longest edges to bisect in a unit simplex refinement. J. Comb. Optim. (2015). https://doi.org/10.1007/s10878-015-9970-y

20. Sherali, H., Adams, W.: A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer Academis Publishers, Dordrecht (1999)
21. Sherali, H., Liberti, L.: Reformulation-Linearization Technique for Global Optimization. In: Encyclopedia of Optimization, Springer, New york (2009)
22. Stolfi, J., de Figueiredo, L.: Self-Validated Numerical Methods and Applications. Monograph for 21st Brazilian Mathematics Colloquium. IMPA/CNPq (1997)
23. Surjanovic, S., Bingham, D.: Virtual library of simulation experiments: Test functions and datasets (2013). http://www.sfu.ca/~ssurjano
24. Todd, M.J.: The Computation of Fixed Points and Applications. Springer, Heidelberg (1976)
25. Tóth, B., Casado, L.G.: Multi-dimensional pruning from the Baumann point in an Interval Global Optimization Algorithm. J. Glob. Optim. **38**(2), 215–236 (2007). https://doi.org/10.1007/s10898-006-9072-6
26. Žilinskas, J.: Branch and bound with simplicial partitions for global optimization. Math. Modell. Anal. **13**(1), 145–159 (2008). https://doi.org/10.3846/1392-6292.2008.13.145-159