J Glob Optim (2007) 38:215–236
DOI 10.1007/s10898-006-9072-6

ORIGINAL PAPER

Multi-dimensional pruning from the Baumann point in an Interval Global Optimization Algorithm

Boglárka Tóth · Leocadio G. Casado

Received: 31 December 2005 / Accepted: 27 July 2006 / Published online: 27 September 2006

Abstract A new pruning method for interval branch and bound algorithms is presented. In reliable global optimization methods there are several approaches to make the algorithms faster. In minimization problems, interval B&B methods use a good upper bound of the function at the global minimum and good lower bounds of the function at the subproblems to discard most of them, but they need efficient pruning methods to discard regions of the subproblems that do not contain global minimizer points. The new pruning method presented here is based on the application of derivative information from the Baumann point. Numerical results were obtained by incorporating this new technique into a basic Interval B&B Algorithm in order to evaluate the achieved improvements.

Keywords Interval arithmetic · Branch-and-bound method · Global optimization · Pruning test

1 Introduction

The problem of finding the global minimum f^* of a real valued n-dimensional continuously differentiable function f: $S \to \mathbb{R}$, $S \subset \mathbb{R}^n$, and the corresponding set S^* of

This work has been supported by the Ministry of Education and Science of Spain through grants TIC 2002-00228, TIN2005-00447, and research project SEJ2005-06273 and by the Integral Action between Spain and Hungary by grant HH2004-0014.
Boglárka Tóth: On leave from the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and the University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary.

B. Tóth (✉)
Departamento de Estadística e Investigación Operativa, Universidad de Murcia,
Campus Universitario de Espinardo, Murcia 30100, Spain.
e-mail: boglarka@um.es

L. G. Casado
Departamento de Arquitectura de Computadores y Electrónica, Universidad de Almería,
Crta. Sacramento s/u, Almería 04120, Spain.
e-mail: leo@ace.ual.es

global minimizers is considered, i.e.,

$$f^* = f(s^*) = \min_{s \in S} f(s),\ s^* \in S^*. \tag{1}$$

The following notation is used. Real numbers are denoted by $x, y, \ldots$ and compact intervals by $X = [x^L, x^U], Y = [y^L, y^U], \ldots$, where $x^L = \min\{x \in X\}$ and $x^U = \max\{x \in X\}$ are the lower and upper bounds of X, respectively. The set of compact intervals is denoted by $\mathbb{I} := \{[a,b] \mid a,b \in \mathbb{R},\ a \le b\}$. The notation $x = (x_1, \ldots, x_n)^T,\ x_i \in \mathbb{R}$ and $X = (X_1, \ldots, X_n)^T,\ X_i \in \mathbb{I}\ (i = 1, \ldots, n)$ is used for real and interval vectors, respectively. The set of n-dimensional interval vectors (also called boxes) is denoted by $\mathbb{I}^n$. The width of the interval X is defined by $w(X) = x^U - x^L$, if $X \in \mathbb{I}$, and $w(X) = \max_{i=1,\ldots,n} w(X_i)$, if $X \in \mathbb{I}^n$. The midpoint of the interval X is defined by $\mathrm{mid}(X) = (x^L + x^U)/2$, for one-dimensional intervals, and $\mathrm{mid}(X) = (\mathrm{mid}(X_1), \mathrm{mid}(X_2), \ldots, \mathrm{mid}(X_n))^T$, for n-dimensional intervals. Let $f\colon Y \subseteq \mathbb{R}^n \to \mathbb{R}$ be a continuous function, and $\mathbb{I}(Y) = \{X \mid X \in \mathbb{I}^n, X \subseteq Y\}$. The function $F\colon \mathbb{I}(Y) \subseteq \mathbb{I}^n \to \mathbb{I}$ is an *inclusion function* of f, if for every $X \in \mathbb{I}(Y)$ and $x \in X, f(x) \in F(X)$, i.e. $f(X) = \{f(x)\ |x\ \in X\} \subseteq F(X)$. Let X be a box and $c \in X$ any point of it. If $G(X)$, i.e. an inclusion function of the gradient vector $g(X)$ is known, then the *centered form* is defined as

$$F_c(X) = F(c) + G(X)(X - c).$$

Usually we use the midpoint ($c = \mathrm{mid}(X)$), but in general, only $c \in X$ is assumed. One interesting choice for c is the Baumann point (b) [1], that is defined by

$$b_i = \begin{cases} \dfrac{g_i^U x_i^L - g_i^L x_i^U}{g_i^U - g_i^L}, & \text{if } g_i^L < 0 < g_i^U \\ x_i^U & \text{if } g_i^U \le 0 \\ x_i^L & \text{if } g_i^L \ge 0 \end{cases} \qquad i = 1, \ldots, n, \tag{2}$$

where the gradient $G(X) = ([g_1^L, g_1^U], \ldots, [g_n^L, g_n^U])$. Baumann proved that ${F_c}^L(X) \le {F_b}^L(X), \forall c \in X$.

In those cases where the objective function $f(x)$ is given by a formula, it is possible to use an interval B&B approach to solve problem (1) (see [9,12,13,15]). A general interval GO (IGO) algorithm based on this approach is shown in Algorithm 1.

Algorithm 1 A general interval B&B GO algorithm

Funct IGO(S, f)

1. Set the working list $L := \{S\}$ and the final list $Q := \{\}$
2. **while** ($L \neq \{\}$)
3. Select an interval X from L — *Selection rule*
4. Compute a lower bound for $f(X)$ — *Bounding rule*
5. **if** X cannot be eliminated — *Elimination rule*
6. Divide X into $X^j,\ j = 1, \ldots, p$, subintervals — *Division rule*
7. **for** j=1,..., p
8. **if** X^j satisfies the termination criterion — *Termination rule*
9. Store X^j in Q
10. **else**
11. Store X^j in L
12. **return** Q

An overview on the theory and the history of the rules of this algorithm can be found, for example, in [9]. Of course, every particular realization of Algorithm 1 depends on the available information on the objective function $f(x)$. In this paper it is supposed that inclusion functions can be evaluated for $f(x)$ and for its first derivative, $g(x)$, over all $X \in \mathbb{I}(S)$. When this information is available, the rules of the traditional realization of Algorithm 1 can be written more precisely. Below we describe a Multidimensional Interval Global Optimization algorithm (MIGO) which is frequently used to solve problem (1) (see [9]).

Selection rule: Among all the intervals X^j stored in the working list L, select an interval X such that $F^L(X) = \min\{F^L(X^j) : X^j \in L\}$.

Bounding rule: The fundamental theorem of interval arithmetic provides a natural and rigorous way to compute an *inclusion function*. In the present study the inclusion function F of the objective function f is available by the naive interval arithmetic and the centered form [6,9]. The interval $f(X)$ is bounded by the interval $F(X) \cap F_c(X)$, i.e. the lower bound for $f(X)$ is $\max\{F^L(X), {F_c}^L(X)\}$.

Elimination rules: Common elimination rules are the following:

- **Midpoint or Cutoff test**: Every interval X is rejected if $F^L(X) > \tilde{f}$, where $\tilde{f}$ is the best known upper bound of f^*. The value of $\tilde{f}$ is usually updated by $F^U(c)$, generally using $c = \text{mid}(X)$.
- **Monotonicity test**: If for an interval X the condition $0 \notin G(X)$ is fulfilled, then this means that the interval X does not contain any minimizer point (the box is rejected) or the minimizer points are on the border of the search region (the box is reduced).

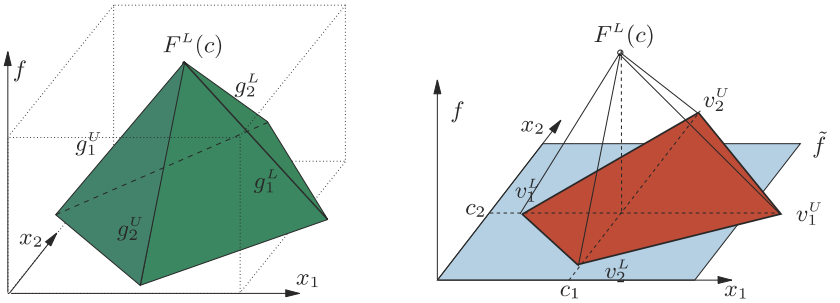
Division rule: Usually two subintervals are generated by bisection using $\text{mid}(X)$ as the subdivision point on direction k, where k is a coordinate such that $w(X_k) = \max_{i=1,\ldots,n} w(X_i)$.

Termination rule: A parameter ε determines the desired accuracy of the solution. Therefore, intervals X with $w(X) \leq \varepsilon$, are moved to the final list Q. Other termination criteria can be found in [15].

In this paper, a new elimination rule based on the application of the gradient information from the Baumann point is proposed. It is in some sense the generalization of the method in [11]. Section 2 shows the idea for the two-dimensional case, while Section 3 extends the theory to the n-dimensional case. Section 4 describes the method algorithmically and Section 5 shows the numerical results and conclusions and finally Section 6 gives a summary.

2 The two-dimensional problem

Before explaining the general method, let us demonstrate it on a two-dimensional problem. Let us examine the visualization of the centered form for a box $X = (X_1, X_2)$. We suppose that $0 \in G(X) = ([g_1^L(X), g_1^U(X)], [g_2^L(X), g_2^U(X)])$ (abbreviated to $([g_1^L, g_1^U], [g_2^L, g_2^U])$), because otherwise the function is monotonic in the box and it could be discarded or reduced. A "tent" can be drawn from the point $(c, F^L(c))$ using the bounds of the derivatives (see Fig. 1(a)). It is easy to see that the function has to be above the tent. If we have a good upper bound on the global minimum ($\tilde{f}$) which is smaller than $F^L(c)$, we know that the minimum cannot be attained in the region defined by the intersection of the tent and the plane $\tilde{f}$. This region is drawn in



(a) The tent. **(b)** The section of the tent and the plane $\tilde{f}$.

Fig. 1 The two-dimensional case

dark(red) in Fig. 1(b), and will be denoted by *PR* (Pruneable Region). The use of this pruneable region was not suggested in [16] because the edges of *PR* are not parallel to the coordinate axes and the division of X by rejecting part of *PR* would generate a lot of boxes. Here it is shown that *PR* is still of interest.

PR can be defined by its vertices:

$$v_1^U = \left\{(x_1, c_2) \mid \tilde{f} = F^L(c) + g_1^L(x_1 - c_1)\right\} = \left(c_1 + \left(\tilde{f} - F^L(c)\right)/g_1^L, c_2\right), \tag{3}$$

$$v_1^L = \left(c_1 + \left(\tilde{f} - F^L(c)\right)/g_1^U, c_2\right), \tag{4}$$

$$v_2^U = \left(c_1, c_2 + \left(\tilde{f} - F^L(c)\right)/g_2^L\right), \tag{5}$$

$$v_2^L = \left(c_1, c_2 + \left(\tilde{f} - F^L(c)\right)/g_2^U\right), \tag{6}$$

if $g_i^L \neq 0$ and $g_i^U \neq 0, i \in \{1,2\}$. Otherwise $v_i^L = -\infty$ or $v_i^U = \infty$ for the appropriate vertex, as the limits of the fractions suggest. Let us introduce the following notation to simplify the formulas.

$$pr_i^U = \frac{\tilde{f} - F^L(c)}{g_i^L}, \quad pr_i^L = \frac{\tilde{f} - F^L(c)}{g_i^U}, \qquad i = 1, 2. \tag{7}$$

Thus $v_1^L = (c_1 + pr_1^L, c_2)$, $v_1^U = (c_1 + pr_1^U, c_2)$ and $v_2^L = (c_1, c_2 + pr_2^L)$, $v_2^U = (c_1, c_2 + pr_2^U)$.

Since our interval B&B algorithm works with boxes, we cannot use other shapes (different from boxes) to divide the non-rejected area, i.e. the remaining region cannot be approximated in the way shown on the left hand side of Fig. 2. On the other hand, if we approximate the non-rejected regions by boxes, too many boxes can be generated, and/or only a small part of the pruneable region can be discarded (see Fig. 2). In general, more than four generated subboxes are not desired because the computational cost of the algorithm can increase accordingly.

One possible goal of the pruning is to obtain the largest box to be removed from the original box. This can be done by computing the largest rectangle in the triangle defined by its vertices v_1^U, v_2^L, c (see Fig. 3). The area of the rectangle is $A = a \cdot b$, where $b = pr_2^U - pr_2^U/pr_1^U a$. Thus, it can be computed by maximizing A with respect to a and b. The maximal area is $pr_1^U pr_2^U/4$ with $a = pr_1^U/2$ and $b = pr_2^U/2$. From the above result one can see that the same can be obtained for all of the four triangles.

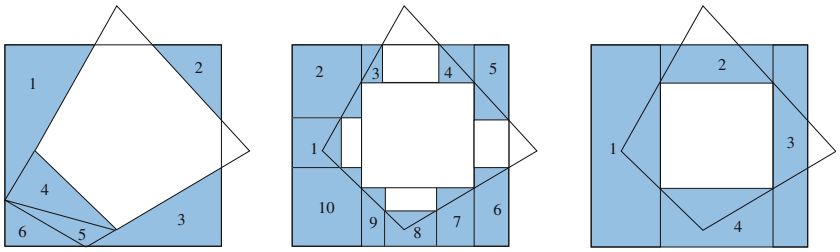


Fig. 2 Examples of the different cutting choices

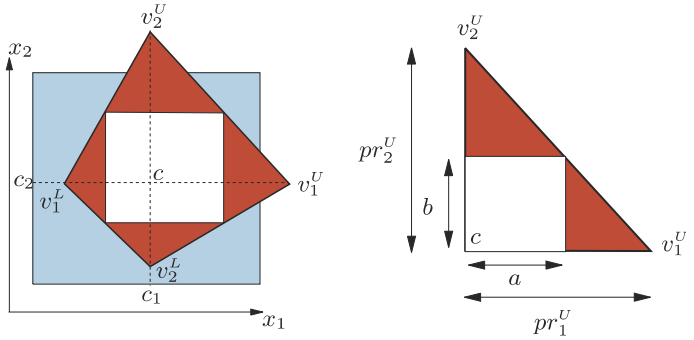


Fig. 3 The largest box which can be cut from *PR*

Fortunately all the rectangles share one edge, thus the resulting box can be given by $c + ([pr_1^L/2, pr_1^U/2], [pr_2^L/2, pr_2^U/2])$. (Note that pr_1^L and pr_2^L are negative.)

Sometimes we can find better choices than the largest box in the pruneable region to be pruned, since the latter is not always inside the box X. To calculate the pruneable box, the only important thing is the intersection between the pruneable region and the original box, independently of their position in the coordinate system. So, to obtain easier formulas and notation let us center the whole problem at the point c, i.e. let us change c to be the origin. Thus, we introduce the following notation.

$$\begin{array}{lll} OB \text{ (Original Box):} & OB = X - c, \\ BPR \text{ (Box containing } PR\text{):} & BPR = ([pr_1^L, pr_1^U], [pr_2^L, pr_2^U]), \\ PB \text{ (Pruneable Box):} & PB = ([pb_1^L, pb_1^U], [pb_2^L, pb_2^U]), \end{array} \tag{8}$$

where BPR is the smallest box which contains the pruneable region (PR) (see Fig. 4). In our new notation the Centered Pruneable Box (CPB) is

$$CPB = ([pr_1^L/2, pr_1^U/2], [pr_2^L/2, pr_2^U/2]) \tag{9}$$

and the area of the CPB is

$$A(CPB) = \left(\frac{pr_1^U}{2} - \frac{pr_1^L}{2}\right)\left(\frac{pr_2^U}{2} - \frac{pr_2^L}{2}\right) = \frac{1}{4}A(BPR),$$

i.e. half of the area of PR, which is half of the area of BPR.

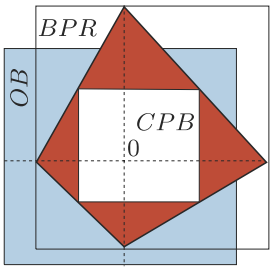


Fig. 4 The new notation for the easier formulation

2.1 Shifting the *CPB* (Centered Pruneable Box)

As it can be seen in Fig. 5, sometimes it is better to shift the *CPB* to the edge of *OB*. This can improve the method by reducing the number of the generated subboxes at the cost of the reduction of the rejected area compared to the area of *CPB*. To distinguish the centered pruneable box (*CPB*) from the shifted pruneable box, the latter will be denoted by *SPB*. In the cases when $OB \cap CPB \neq CPB$, the area of *SPB* can be larger than the area of $CPB \cap OB$ (see the third case in Fig. 5).

Let us suppose that we want to shift *CPB* to the upper side of *OB*, as $SPB_{ob_2^U}$ in Fig. 6. This is possible only if $ob_2^U < pr_2^U$, and therefore $spb_2^U = ob_2^U$ can be obtained. We only have to determine the box coordinates $(spb_1^L, spb_1^U, spb_2^L, spb_2^U)$ (note that spb_1^L, spb_2^L are negative since the origin is inside the box). It is easy to see that the corner (spb_1^U, spb_2^U) is on the edge of the pruneable region, iff

$$1 = \frac{spb_1^U}{pr_1^U} + \frac{spb_2^U}{pr_2^U}.$$

We know that $spb_2^U = ob_2^U$, thus $spb_1^U = pr_1^U \left(1 - \frac{ob_2^U}{pr_2^U}\right)$. For the other corners

$$1 = \frac{spb_1^U}{pr_1^U} + \frac{spb_2^L}{pr_2^L}, \qquad 1 = \frac{spb_1^L}{pr_1^L} + \frac{spb_2^U}{pr_2^U}, \qquad \text{and} \qquad 1 = \frac{spb_1^L}{pr_1^L} + \frac{spb_2^L}{pr_2^L}$$

have to hold, thus

$$spb_1^L = pr_1^L \left(1 - \frac{ob_2^U}{pr_2^U}\right), \quad spb_2^L = pr_2^L \frac{ob_2^U}{pr_2^U}.$$

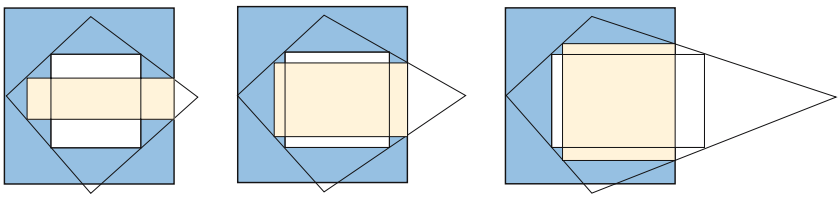


Fig. 5 Shifting the pruneable box to an edge of *OB*

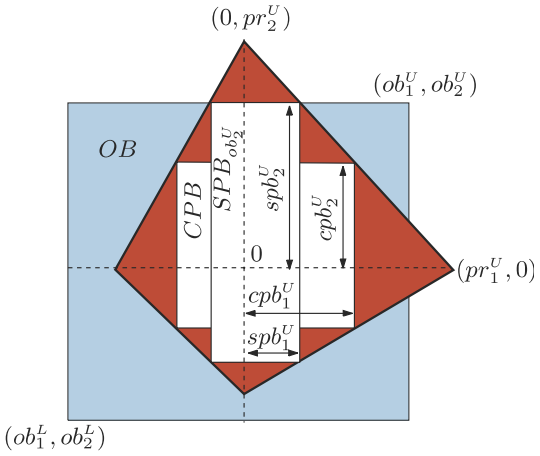


Fig. 6 Shifting example

Now we have a new problem to solve: which side to choose for shifting, and moreover whether it is worth shifting, or not. Both questions can be answered by comparing the areas that can be pruned in each case.

The areas of the boxes—supposing that the resulting *SPB* is inside of the original one (*OB*)—can be computed as

$$\begin{aligned} A(SPB_{ob_2^U}) &= \left(pr_1^U\left(1-\frac{ob_2^U}{pr_2^U}\right)-pr_1^L\left(1-\frac{ob_2^U}{pr_2^U}\right)\right)\left(ob_2^U-pr_2^L\frac{ob_2^U}{pr_2^U}\right)\\ &= \left(pr_1^U-pr_1^L\right)\left(1-\frac{ob_2^U}{pr_2^U}\right)\frac{ob_2^U}{pr_2^U}\left(pr_2^U-pr_2^L\right)\\ &= \left(1-\frac{ob_2^U}{pr_2^U}\right)\frac{ob_2^U}{pr_2^U}A(BPR) \end{aligned} \tag{10}$$

that is, in general

$$A(SPB_{ob_i^I}) = \left(1-\frac{ob_i^I}{pr_i^I}\right)\frac{ob_i^I}{pr_i^I}A(BPR), \quad i=1,2, \quad I=L,U. \tag{11}$$

If an *SPB* is inside the original box then the shifted area only depends on the value $\frac{ob_i^I}{pr_i^I}, i=1,2, I=L,U$, obtaining a larger area if it is nearer 1/2. If it equals to 1/2 we obtain that $A(SPB_{ob_i^I}) = A(CPB)$, which also means that $SPB_{ob_i^I} = CPB$.

These equations show that knowing the vector $\left(\frac{ob_1^U}{pr_1^U}, \frac{ob_1^L}{pr_1^L}, \frac{ob_2^U}{pr_2^U}, \frac{ob_2^L}{pr_2^L}\right)$ one can choose the largest Shifted Pruneable Box. If the largest *SPB* is inside the original box one should only decide to prune this or the *CPB*. The other cases, when one or more corners of the box are inside the pruneable region, do not differ too much, but those have to be treated differently. This would lead to a case analysis that is out of the scope of this work and cannot be extended for the multidimensional case easily.

2.2 Simplification using the Baumann point

In this section, we will see that the use of the Baumann point [1] instead of the center c simplifies our computations, avoiding the previous case analysis.

Theorem 1 *Consider a given differentiable function $f\colon \mathbb{R}^2 \to \mathbb{R}$, its inclusion function F and the inclusion function of its derivative G over a given box X. The pruneable region defined by its vertices, as in equations (3) to (6), centered in the Baumann point (i.e. $c = b$) include either all the corners of X or none of them.*

Proof First we show that in the case when one corner is exactly on the edge of the pruneable region, the other corners are also on the edges of the pruneable region (see Fig. 7). That is, if x^L is such that

$$1 = \frac{x_1^L - b_1}{pr_1^L} + \frac{x_2^L - b_2}{pr_2^L}, \tag{12}$$

then

$$1 = \frac{x_1^U - b_1}{pr_1^U} + \frac{x_2^L - b_2}{pr_2^L}, \quad 1 = \frac{x_1^L - b_1}{pr_1^L} + \frac{x_2^U - b_2}{pr_2^U}, \quad 1 = \frac{x_1^U - b_1}{pr_1^U} + \frac{x_2^U - b_2}{pr_2^U}. \tag{13}$$

It is easy to see that Eq. (12) and (13) hold if and only if

$$\frac{x_1^U - b_1}{pr_1^U} = \frac{x_1^L - b_1}{pr_1^L}, \qquad \frac{x_2^U - b_2}{pr_2^U} = \frac{x_2^L - b_2}{pr_2^L}. \tag{14}$$

Recalling the definitions of $pr_i^U, pr_i^L, i = 1,2$ (see (7)) and b_1, b_2 (see (2)), the equations are

$$\frac{x_i^U - \frac{g_i^U x_i^L - g_i^L x_i^U}{g_i^U - g_i^L}}{\frac{\tilde{f} - F^L(b)}{g_i^L}} = \frac{x_i^L - \frac{g_i^U x_i^L - g_i^L x_i^U}{g_i^U - g_i^L}}{\frac{\tilde{f} - F^L(b)}{g_i^U}}, \qquad i = 1,2$$

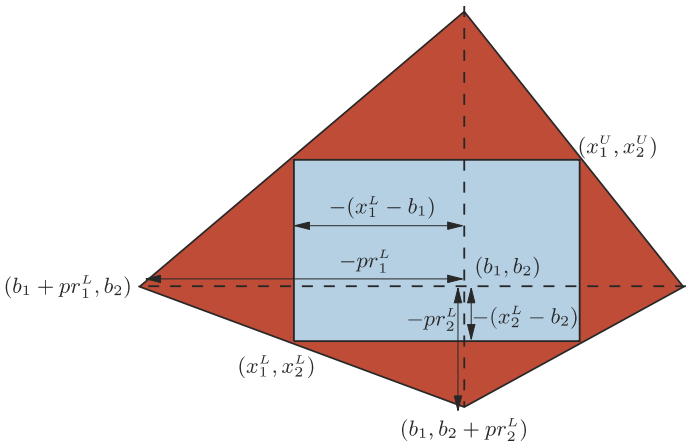


Fig. 7 Example of the case when Baumann point is used

that can be simplified to

$$g_i^L x_i^U - g_i^L \frac{g_i^U x_i^L - g_i^L x_i^U}{g_i^U - g_i^L} = g_i^U x_i^L - g_i^U \frac{g_i^U x_i^L - g_i^L x_i^U}{g_i^U - g_i^L}, \quad i = 1,2,$$

and finally to

$$g_i^L x_i^U - g_i^U x_i^L = \frac{-g_i^U (g_i^U x_i^L - g_i^L x_i^U) + g_i^L (g_i^U x_i^L - g_i^L x_i^U)}{g_i^U - g_i^L}, \quad i = 1,2,$$

which reduces to an identity. Consequently, the Eq. (14) are always true. Thus, if we change the equality in (12) to inequality, then inequality will occur in (13), which proves the theorem. □

Remark 1 The theorem holds also if $g_i^L = 0$ or $g_i^U = 0$ for $i \in \{1,2\}$. In this case as $pr_i^U = \infty$ and $b_i = x_i^L$ or $pr_i^L = -\infty$ and $b_i = x_i^U$,

$$\frac{x_i^U - b_i}{pr_i^U} = \frac{x_i^L - b_i}{pr_i^L} = 0.$$

Corollary 1 *Since* $\forall i \frac{x_i^U - b_i}{pr_i^U} = \frac{x_i^L - b_i}{pr_i^L}$, *it can be deduced easily that the shifted pruneable box* $SPB_{ob_i^U}$ *is exactly the same as* $SPB_{ob_i^L}$. *Thus, it can be denoted as* SPB_i. *Therefore, after shifting there will be only two newly generated boxes* (*the enclosures of the non-rejected areas*), *and only the better shifting direction has to be determined.*

Remark 2 The above results suggest the notation of the Shifting Factor

$$sf_i = \frac{x_i^U - b_i}{pr_i^U} \left(= \frac{x_i^L - b_i}{pr_i^L} \right), \text{i.e. } sf_i = \frac{ob_i^U}{pr_i^U} \left(= \frac{ob_i^L}{pr_i^L} \right), \quad i = 1,2,$$

and the Opposite Shifting Factor

$$osf_i = 1 - \frac{x_i^U - b_i}{pr_i^U}, \text{i.e. } osf_i = 1 - \frac{ob_i^U}{pr_i^U}, \quad i = 1,2.$$

Therefore, $A(SPB_i) = sf_i osf_i A(BPR)$.

Theorem 2 *If* $CPB \not\subseteq OB$ *and* $OB \not\subseteq CPB$, *then there exists a SPB such that* $A(SPB) > A(CPB \cap OB)$.

Proof If $CPB \not\subseteq OB$, then there exists an $i \in \{1,2\}$ such that $sf_i < 1/2$, i.e. in the ith direction $cpb_i^U > ob_i^U$ and $cpb_i^L < ob_i^L$. As $OB \not\subseteq CPB$, for the other direction $j \neq i$, $cpb_j^U < ob_j^U$ and $cpb_j^L > ob_j^L$. The area of the region that can be pruned using the CPB is

$$A(CPB \cap OB) = \frac{1}{2} \frac{ob_i^U - ob_i^L}{pr_i^U - pr_i^L} A(BPR),$$

while the area of SPB_i is

$$A(SPB_i) = sf_i osf_i A(BPR).$$

One can easily see that

$$A(CPB \cap OB) < A(SPB_i) \iff \frac{1}{2}\frac{ob_i^U - ob_i^L}{pr_i^U - pr_i^L} < sf_i osf_i$$
$$\iff \frac{1}{sf_i}\frac{1}{2}\frac{ob_i^U - ob_i^L}{pr_i^U - pr_i^L} < osf_i \iff \frac{1}{2}\frac{\frac{ob_i^U}{sf_i} - \frac{ob_i^L}{sf_i}}{pr_i^U - pr_i^L} < osf_i$$
$$\iff \frac{1}{2}\frac{pr_i^U - pr_i^L}{pr_i^U - pr_i^L} < osf_i \iff \frac{1}{2} < osf_i$$

which is assured by $sf_i < 1/2$. □

Remark 3 In Theorem 2 if $SPB_i \not\subseteq OB$, then $OB \subset SPB_i$. Therefore, the whole box can be deleted. It also means that although the computed area ($A(SPB_i)$) is larger than the possible pruned one, we prune the maximal area, $A(CPB \cap OB) < A(OB) = A(SPB_i \cap OB) < A(SPB_i)$.

The advantages of the usage of the Baumann point are that it makes our computation easier, and it equilibrates the pruneable region over the box.

3 The *n*-dimensional case

To generalize the above results for the multi-dimensional case, let us suppose that $c = (c_1, c_2, \ldots, c_n)$ is an arbitrary point of the n-dimensional box X. Before any computations are made, let us center the problem at the point c as we did in the two-dimensional case. Thus, we will use the same notation:

$$OB \text{ (Original Box): } OB = X - c, \tag{15}$$

It is easy to see, that the vertices of the pruneable region are

$$\begin{aligned} v_i^L &= (0, \ldots, 0, pr_i^L, 0, \ldots, 0), \\ v_i^U &= (0, \ldots, 0, pr_i^U, 0, \ldots, 0), \quad i = 1, \ldots, n, \end{aligned} \tag{16}$$

where

$$pr_i^U = \frac{\tilde{f} - F^L(c)}{g_i^L}, \quad pr_i^L = \frac{\tilde{f} - F^L(c)}{g_i^U}, \quad i = 1, \ldots, n.$$

From (16) we know that in the three-dimensional case the shape of PR is as it is shown in Fig. 8. To see the properties of this body, let us take a little side-track into geometry. Let us first introduce the notion of orthopeder which appears to be a new class of geometrical objects.

Definition 1 An n-dimensional polytope is called *orthopeder*, if the diagonals intersect in one point and are orthogonal to each other.

Proposition 1 *The PR defined by its vertices* (16) *is an orthopeder.*

Proof The diagonals of the PR are the lines between $v_i^L, v_i^U, i = 1, \ldots, n$, i.e. between the non-adjacent vertices (16). As all the lines pass through the origin, this is the intersection point. The ith diagonal v_i^L, v_i^U is parallel to the ith axis for all i, thus all the diagonals are orthogonal. □

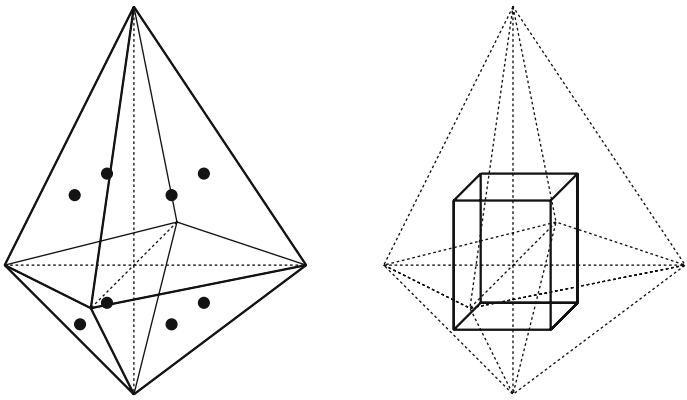


Fig. 8 The shape of *PR* in the three-dimensional case

Definition 2 The *cross polytope* is the regular polytope in n dimensions corresponding to the convex hull of the points formed by permuting the coordinates $(\pm 1, 0, 0, \ldots, 0)$.

An orthopeder is a generalization of a cross polytope, with different scaling on all directions. It is easy to see, that the corresponding scaling vectors are $-PR^L$ and PR^U. The cross polytope is a dual of an n-dimensional hypercube, i.e. its vertices are the centers of the faces of the hypercube. As it is known in geometry, in the cross polytope the largest hyperrectangle is its dual, i.e. the hypercube which has its vertices on the center of the faces of the cross polytope.

Proposition 2 *It is trivial that in an orthopeder* (*a scaled cross polytope*) *the largest hyperrectangle is the rescaled hypercube with the same scale vectors.*

Corollary 2 *The largest box in the orthopeder PR defined by its vertices* (16) *is the Centered Pruneable Box (CPB),*

$$CPB = \left(\left[pr_1^L/n, pr_1^U/n,\right], \ldots, \left[pr_n^L/n, pr_n^U/n,\right]\right).$$

Corollary 3 *The volume of CPB is*

$$V(CPB) = \prod_{i=1,\ldots,n} \frac{pr_i^U - pr_i^L}{n} = \frac{1}{n^n} V(BPR).$$

Theorem 3 *Let* $f\colon \mathbb{R}^n \to \mathbb{R}$ *be a given differentiable function, F its inclusion function and G the inclusion function of its derivative over a given box X. The pruneable region defined by its vertices as in* (16) *centered in the Baumann point* (*i.e.* $c = b$) *includes either all the corners of X or none of them.*

Proof It can be proved in the same way like Theorem 1. We first prove that when one corner of X is exactly on one face of the pruneable region, the other corners are also on the corresponding faces of the pruneable region. For this we will use the notation $J^i = (J_1^i, \ldots, J_n^i); J_k^i \in \{L, U\}, k = 1, \ldots, n$, where the ith vertex of X is $w_i = (x_1^{J_1^i}, \ldots, x_n^{J_n^i})$.

If w_i is such that

$$1 = \sum_{k=1}^{n} \frac{x_k^{j_k^i} - b_k}{pr_k^{j_k^i}}, \tag{17}$$

then

$$1 = \sum_{k=1}^{n} \frac{x_k^{j_k^j} - b_k}{pr_k^{j_k^j}}, \quad \forall j = 1, \ldots, 2^n. \tag{18}$$

It is easy to see that equations (18) can hold iff

$$\frac{x_k^{j_k^i} - b_k}{pr_k^{j_k^i}} = \frac{x_k^{j_k^j} - b_k}{pr_k^{j_k^j}} \quad \forall j = 1, \ldots, 2^n;\ k = 1, \ldots, n,$$

i.e.

$$\frac{x_k^U - b_k}{pr_k^U} = \frac{x_k^L - b_k}{pr_k^L}, \quad k = 1, \ldots, n. \tag{19}$$

As the Baumann point is defined piecewise, the proof of Eq. (19) does not differ from the demonstration of Eq. (14) which appears in the proof of Theorem 1.

Equations (19) are always true. Thus, changing the equality in (17) to inequality, the same inequality change occurs in (18). It means that when one vertex of X is inside (or outside) PR, the other vertices are also inside (or outside), which proves the theorem. □

3.1 Shifting the n-dimensional *CPB*

When *CPB* is inside *OB* and *CPB* is pruned, the number of generated boxes is $2n$. If $n > 3$ the performance of the algorithm can decrease because more boxes have to be evaluated. Therefore, it is important to find the best direction(s) to shift.

To choose the best *PB*, we are going to construct a simple and powerful method that determines the *SPB* with the *largest volume/fewer new boxes* index. We already know that using the Baumann point the equations

$$\frac{ob_i^U}{pr_i^U} = \frac{ob_i^L}{pr_i^L}, \quad i = 1, \ldots, n$$

hold (which are equivalent to (19)). In the following formulas these values will appear many times, thus, let us introduce the notation of Shifting Factor (sf_i) and Opposite Shifting Factor (osf_i) as we did in the 2D case

$$sf_i = \frac{ob_i^U}{pr_i^U} = \frac{ob_i^L}{pr_i^L}, \quad osf_i = 1 - sf_i, \quad i = 1, \ldots, n,$$

Shifting in the dimension i is possible only if $sf_i < 1$ ($osf_i > 0$), i.e. when $ob_i^L > pr_i^L$ and $ob_i^U < pr_i^U$.

The corner $(spb_1^U, \ldots, spb_n^U)$ is on the face of the pruneable region, iff

$$1 = \frac{spb_1^U}{pr_1^U} + \frac{spb_2^U}{pr_2^U} + \cdots + \frac{spb_n^U}{pr_n^U}.$$

Let us suppose that $spb_j^U = ob_j^U$ and so $spb_j^L = ob_j^L$, i.e. we shift in the jth dimension. Therefore, if the other coordinates of CPB have to be scaled with the same factor r, i.e. $spb_i^I = r \cdot cpb_i^I = r\frac{pr_i^U}{n}$, $i = 1, \ldots, n, i \neq j$, then

$$1 = \frac{ob_j^U}{pr_j^U} + \sum_{\substack{i=1,\ldots,n \\ i\neq j}} \frac{r\frac{pr_i^U}{n}}{pr_i^U}, \qquad r = \frac{n}{n-1}\left(1 - sf_j\right) = \frac{n}{n-1} osf_j,$$

$$spb_i^I = \frac{pr_i^I}{n-1} osf_j, \qquad i = 1, \ldots, n \quad I = L, U.$$

Thus, if SPB_j is inside OB, its volume can be computed as

$$\begin{aligned} V(SPB_j) &= \left(ob_j^U - ob_j^L\right) \prod_{\substack{i=1,\ldots,n \\ i\neq j}} \left(\frac{pr_i^U}{n-1} osf_j - \frac{pr_i^L}{n-1} osf_j\right) \\ &= \left(ob_j^U - \frac{pr_j^L}{pr_j^U} ob_j^U\right)\left(\frac{n}{n-1} osf_j\right)^{n-1} \prod_{\substack{i=1,\ldots,n \\ i\neq j}} \frac{pr_i^U - pr_i^L}{n} \\ &= n\left(\frac{n}{n-1}\right)^{n-1} \frac{ob_j^U}{pr_j^U}\, osf_j^{n-1} \prod_{i=1,\ldots,n} \frac{pr_i^U - pr_i^L}{n} \\ &= \frac{n^n}{(n-1)^{n-1}}\, sf_j\, osf_j^{n-1}\, V(CPB). \end{aligned}$$

One can see, that the nearer sf_j is $1/n$, the greater the volume is. It also implies that knowing the vector $(sf_1, \ldots, sf_n)$ we can easily choose the best direction to shift.

In some cases it is possible to shift in several dimensions. To shift in $j_1, \ldots, j_k \in \{1, \ldots, n\}$ dimensions, we generalize the notation of the Opposite Shifting Factor: $osf_{j_1,\ldots,j_k} = 1 - \sum_{i=j_1,\ldots,j_k} sf_i$. Shifting is possible only if $osf_{j_1,\ldots,j_k} > 0$. Thus, we can compute the coordinates of the $SPB_{j_1,\ldots,j_k}$ in the following way:

$$\begin{aligned} 1 &= \sum_{i=j_1,\ldots,j_k} \frac{ob_i^U}{pr_i^U} + \sum_{\substack{i=1,\ldots,n \\ i\neq j_1,\ldots,j_k}} \frac{r\frac{pr_i^U}{n}}{pr_i^U}, \\ r &= \frac{n}{n-k}\left(1 - \sum_{i=j_1,\ldots,j_k} sf_i\right) = \frac{n}{n-k}\, osf_{j_1,\ldots,j_k}, \\ spb_i^I &= \frac{pr_i^I}{n-k}\, osf_{j_1,\ldots,j_k}, \qquad i = 1, \ldots, n \quad I = L, U. \end{aligned} \tag{20}$$

The volume of $SPB_{j_1,\ldots,j_k}$ if it is inside OB is the following

$$
\begin{aligned}
&V(SPB_{j_1,\ldots,j_k}) \\
&= \prod_{i=j_1,\ldots,j_k} \left(ob_i^U - ob_i^L\right) \prod_{\substack{i=1,\ldots,n \\ i\neq j_1,\ldots,j_k}} \left(\frac{pr_i^U}{n-k}\, osf_{j_1,\ldots,j_k} - \frac{pr_i^L}{n-k}\, osf_{j_1,\ldots,j_k}\right) \\
&= \prod_{i=j_1,\ldots,j_k} \left(ob_i^U - \frac{pr_i^L}{pr_i^U} ob_i^U\right) \left(\frac{n}{n-k}\, osf_{j_1,\ldots,j_k}\right)^{n-k} \prod_{\substack{i=1,\ldots,n \\ i\neq j_1,\ldots,j_k}} \frac{pr_i^U - pr_i^L}{n} \\
&= n^k \prod_{i=j_1,\ldots,j_k} \frac{ob_i^U}{pr_i^U} \left(\frac{n}{n-k}\right)^{n-k} osf_{j_1,\ldots,j_k}^{n-k} \prod_{i=1,\ldots,n} \left(pr_i^U - pr_i^L\right)/n \\
&= \frac{n^n}{(n-k)^{n-k}} \left(\prod_{i=j_1,\ldots,j_k} sf_i\right) osf_{j_1,\ldots,j_k}^{n-k} V(CPB).
\end{aligned}
$$

Remark 4 If we have the volume for SPB, shifted in the dimensions $j_1,\ldots,j_{k-1} \in \{1,\ldots,n\}$, and we want to shift it in one more dimension, $j_k \in \{1,\ldots,n\}$ as well, the new volume can be computed from the known $V(SPB_{j_1,\ldots,j_{k-1}})$ in the following way:

$$
V(SPB_{j_1,\ldots,j_k}) = sf_{j_k} \frac{(n-k+1)^{n-k+1}}{(n-k)^{n-k}} \frac{osf_{j_1,\ldots,j_k}^{n-k}}{osf_{j_1,\ldots,j_{k-1}}^{n-k+1}} V(SPB_{j_1,\ldots,j_{k-1}}).
$$

Similarly to Theorem 2, Theorem 4 shows that if CPB is not inside OB we can always obtain an SPB with larger volume than $CPB \cap OB$.

Theorem 4 *Let us suppose $CPB \not\subseteq OB$ in the dimensions $j_1,\ldots,j_k \in \{1,\ldots,n\}$, ($k < n$), i.e. $ob_i^U < cpb_i^U$ and $ob_i^L > cpb_i^L$ for all $i \in \{j_1,\ldots,j_k\}$, but not for $i \in \{1,\ldots,n\} \setminus \{j_1,\ldots,j_k\}$. Then $V(SPB_{j_1} \cap OB) > V(CPB \cap OB)$, and $V(SPB_{j_1,\ldots,j_l}) > V(SPB_{j_1,\ldots,j_{l-1}} \cap OB), l \leq k$.*

Proof First we will prove that $V(SPB_{j_1} \cap OB) > V(CPB \cap OB)$. From the assumptions one can obtain that $sf_{j_1} < 1/n$, and so $osf_{j_1} > (n-1)/n$.

The volume of the region which can be pruned using CPB is

$$
V(CPB \cap OB) = \frac{1}{n^{n-k}} \prod_{i=j_1,\ldots,j_k} \frac{ob_i^U - ob_i^L}{pr_i^U - pr_i^L} V(BPR).
$$

The volume of the region which can be pruned using SPB_{j_1} is

$$
\begin{aligned}
V(SPB_{j_1} \cap OB) &= \prod_{\substack{i=1,\ldots,n \\ i\neq j_1,\ldots,j_k}} \frac{osf_{j_1}}{n-1} \left(pr_i^U - pr_i^L\right) \prod_{i=j_1,\ldots,j_k} \left(ob_i^U - ob_i^L\right) \\
&= \frac{osf_{j_1}^{n-k}}{(n-1)^{n-k}} \prod_{i=j_1,\ldots,j_k} \frac{ob_i^U - ob_i^L}{pr_i^U - pr_i^L} V(BPR).
\end{aligned}
$$

Thus,

$$V(CPB \cap OB) < V(SPB_{j_1} \cap OB) \iff \frac{1}{n^{n-k}} < \frac{1}{(n-1)^{n-k}} osf_{j_1}^{n-k}$$
$$\iff \left(\frac{n-1}{n}\right)^{n-k} < osf_{j_1}^{n-k} \iff \frac{n-1}{n} < osf_{j_1},$$

what is assured by $sf_{j_1} < 1/n$.

Let us suppose now, that we have already shifted in $j_1, \ldots, j_{l-1}$ directions, but $SPB_{j_1,\ldots,j_{l-1}} \not\subseteq OB$. We know that

$$V(SPB_{j_1,\ldots,j_l} \cap OB) = \frac{osf_{j_1,\ldots,j_l}^{n-k}}{(n-l)^{n-k}} \prod_{i=j_1,\ldots,j_k} \frac{ob_i^U - ob_i^L}{pr_i^U - pr_i^L} V(BPR).$$

$$\begin{aligned}
V(SPB_{j_1,\ldots,j_{l-1}} \cap OB) &< V(SPB_{j_1,\ldots,j_l} \cap OB) \\
\iff \frac{osf_{j_1,\ldots,j_{l-1}}^{n-k}}{(n-l+1)^{n-k}} &< \frac{osf_{j_1,\ldots,j_l}^{n-k}}{(n-l)^{n-k}} \\
\iff (n-l) osf_{j_1,\ldots,j_{l-1}} &< (n-l+1) osf_{j_1,\ldots,j_l} \\
\iff (n-l) osf_{j_1,\ldots,j_{l-1}} &< (n-l+1)(osf_{j_1,\ldots,j_{l-1}} - sf_{j_l}) \\
\iff (n-l+1) sf_{j_l} &< osf_{j_1,\ldots,j_{l-1}} \\
\iff (n-l+1) sf_{j_l} + \sum_{i=j_1,\ldots,j_{l-1}} sf_i &< 1.
\end{aligned}$$

We know that $sf_i < 1/n,\ i = j_1, \ldots, j_k$. Thus, the above equation holds, and therefore $V(SPB_{j_1,\ldots,j_{l-1}} \cap OB) < V(SPB_{j_1,\ldots,j_l} \cap OB)$ if $l \leq k$. □

Corollary 4 *The largest pruneable box inside OB is $SPB_{j_1,\ldots,j_k}$ if $ob_i^U < cpb_i^U$ for all $i \in \{j_1, \ldots, j_k\}$, and only for those.*

4 Integrating the pruning method into the interval B&B algorithm

The Baumann Tent Pruning Dividing (BTPD) method can be incorporated into Algorithm 1 by changing the Division Rule in line 6 by a call to Algorithm 2.

The BTPD method will prune-divide the current box or just divide it depending on several factors. Firstly, a Pruning Index (*PI*) of the best *PB*, i.e. the *relative volume/number of new boxes* is calculated. If $PI \cdot n^n$ is smaller than a given input parameter α, the normal division rule is applied, avoiding the unpromising pruning of *PB* (see Algorithm 2 line 11). The multiplication by n^n comes from the fact that the volume of *CPB* is n^n-part of the volume of *BPR*, thus, to give a shifted *PB* a chance to be pruned, we multiply the Pruning Index by n^n. Secondly, if the box is badly shaped, i.e. the ratio between the minimal and maximal width is less than a parameter β, we just divide it by the normal division rule in order to avoid needle shapes. Finally, the third condition is to use the normal division rule if the minimal width of the box is less than ε in order to avoid the division of the dimensions in which the termination criterion are fulfilled. The second and third conditions are controlled in Algorithm 3 which returns -1 if either of them is satisfied (See Algorithm 2 lines 8 and 9).

If the previous conditions are not satisfied, Algorithm 2 will prune the *PB* from the current box (and so divide it too) if $PI \cdot n^n > \alpha$ (See Algorithm 2, lines 11–21).

In steps 2–3 of Algorithm 2 we check whether the box is inside the pruneable region in order to discard the box if possible. Otherwise, if *PR* is inside the box, only *CPB* can be pruned, and so *PI* (Pruning Index) is computed by the *relative volume/number of new boxes*.

If *CPB* can be shifted, we call the ChoosePB procedure (See Algorithm 3) which returns the number of shifted directions, the best *PB*, and the value of *PI*.

Algorithm 2 The Baumann Tent Pruning Division method (BTPD)

Funct BTPD$(X, GX, b, F(b), \tilde{f}, \alpha)$

1. $OB = X - b; PR = \frac{\tilde{f} - F^L(b)}{GX}$
2. in $= 1 - \sum_{i=1}^{n} \frac{OB_i^L}{PR_i^L}$
3. **if** (in > 0) *The box is inside the pruneable region,*
4. **return** *so we can discard the whole box*
5. **else if** $(PR \subseteq OB)$
6. $PB = PR/n; PI = Vol(PB)/(2n \cdot Vol(OB)); nshift = 0;$
7. **else**
8. $nshift$ = ChoosePB(OB, PR, PB, PI)
9. **if** ($nshift$ == -1) *The box is badly shaped and the new*
10. DivideBox(X); **return** *subboxes would inherit it*
11. **if** $(PI \cdot n^n > \alpha)$ *It is worth pruning*
12. **for** $(i = 1; i \leq n; i++)$
13. $d_i = \min\{OB_i^L - PB_i^L, PB_i^U - OB_i{}^U\}$ *To obtain more cube like*
14. $sort = Sort(d); Y = X$; j=1; *boxes the widths of the fixed*
15. **for** $(k = 1; k \leq n; k++)$ *new sides are ordered.*
16. $i = sort_k;$
17. **if** $(c_i + PB_i^L > X_i^L)$
18. $X^j = Y; X_i^j = [X^L, c_i + PB_i^L]$; j++ *Generating*
19. **if** $(c_i + PB_i^U < X_i^U)$ *new boxes*
20. $X^j = Y; X_i^j = [c_i + PB_i^U, X^U]$; j++
21. $Y_i = c_i + PB_i;$
22. **else**
23. DivideBox(X); **return**

Procedure ChoosePB decides which directions should be shifted in order to obtain the best *PB*. It starts with a call to GetShiftFactorAndVol procedure (see Algorithm 4) where the relative volume vol_0 is calculated by shifting only in the dimensions where *CPB* is outside the current box. *PI* of *SPB* and the shifting factors are also calculated. The volume vol_0 is the maximal relative volume of any *PB* (see Corollary 4). Variable *out* returns the number of directions where *CPB* was outside the current box and therefore shifted.

Algorithm 3 checks if more shifting is possible. Thus, the *sf* vector is sorted in increasing order. Since the first *out* directions are already shifted, only the remaining ones are checked (see lines 7–13). Step 9 computes a new relative volume from the previous one (see Remark 4). If the *relative volume/number of new boxes* is better than the previous *PI*, we update it. Finally, Steps 15–18 compute the best *PB*, and return. If the best *PB* has a high enough *PI*, Algorithm 2 computes the new boxes in Steps 12–21 in a way that more cube like boxes are generated.

Algorithm 3 Choose the best PB

Funct ChoosePB(OB, PR, PB)

1. $n = Dimension(OB)$;
2. **if** (!GetShiftFactorAndVol(OB, PR, sf, vol_0, out)) *The box is badly shaped*
3. **return** -1; *and the new boxes would inherit it*
4. $sort = Sort(-sf)$;
5. $nshift = out$; *nshift will be the number of directions to shift*
6. $PI = vol_0/(2n - 2out)$; *PI is the pruning index*
7. **for** $(k = out + 1; k < n; k++)$
8. **if** $(osf - sf(sort_k) < 0)$ **break** ; *We cannot shift more*
9. $vol_k = vol_{k-1} \frac{sf(sort_k)}{osf} \left(\frac{osf - sf(sort_k)}{osf} \right)^{n-k} \frac{(n-k+1)^{n-k+1}}{(n-k)^{n-k}}$;
10. **if** $(vol_k/(2n - 2k) \geq PI)$
11. $osf- = sf(sort_k)$;
12. $PI = vol_k/(2n - 2k)$;
13. $nshift = k$;
14. **else break** ;
15. **for** $(i = 1; i <= nshift; i++)$ *In the shifted dimensions*
16. $PB(sort_i) = OB(sort_i)$; $PB_i = OB_i$,
17. **for** $(i = nshift + 1; i \leq n; i++)$ *otherwise*
18. $PB(sort_i) = osf * PR(sort_i)/(n - nshift)$; *use* (20).
19. **return** *nshift*;

Algorithm 4 Compute the vector sf and the maximal volume of SPB

Funct GetShiftFactorAndVol(OB, PR, sf, vol_0, out)

1. $w_{min} = min\{w(OB_1), \ldots, w(OB_n)\}$; *The minimal width*
2. $osf = 1.0; vol_0 = 1.0$; *vol_0 is the maximal relative volume of any PB*
3. **for** $(i = 1; i \leq n; i++)$
4. **if** $(w_{min}/w(OB_i) > \beta$ && $w(OB_i) > \varepsilon)$ *It is not badly shaped*
5. $sf_i = ob_i^L/pr_i^L$;
6. **if** $(sf_i > 1/n)$
7. $vol_0* = w(PR_i/n)/w(OB_i)$;
8. **else** *We shift the dimensions*
9. $out++$; *where CPB is outside by default*
10. **if** $(pr_i^U > -pr_i^L)$
11. $vol_0* = ob_i^U(1 - pr_i^L/pr_i^U)$; *To avoid overflow*
12. **else** $vol_0* = ob_i^L(pr_i^U/pr_i^L - 1)$; *and underflow*
13. $osf- = sf(i)$;
14. **else** *It is badly shaped*
15. **if** $(w(PR_i) == \infty)$ *we cannot divide in this direction using any PB*
16. **return** 0;
17. **else** $sf_i = 100$; *To avoid shifting in this direction*
18. **if** $(out > 0)$ *If there are already shifted dimensions*
19. $vol_0* = (n/(n - out))^{n-out}$;
20. **return** 1;

We used a naive grid search algorithm to obtain good values for the parameters α and β of the algorithm. In fact, the search was performed in only one parameter at a time, and we used a loop to optimize both parameters in turns until it converges to a local optima. The optimization was done for two objectives: for the total effort (summing up the number of evaluations for all the problems) and for the average efficiency rate (by efficiency rate we mean the number of evaluations of the Algorithm MIGO divided by the the number of evaluations of the Algorithm MIGO with the

BTPD for a problem). In the first case $\alpha = 1.45$ and $\beta = 0.028$ while in the second case $\alpha = 0.38$ and $\beta = 0.027$. The results for the "badly shaped" parameter β are very similar in both cases. This happens because this parameter has an effect only on a few instances. However, the parameter *PI* is quite different, because most of the problems can be solved very easily, while only a few problems take more time and computational effort. Thus, when optimizing the total number of evaluations the fast problems do not really count in the result. This fact is bad news because one cannot decide the best value for a new problem beforehand. On the other hand, the results are not so different for both settings, as one can see it in the next section.

5 Results and conclusions

Experiments have been carried out with a dual processor Intel Xeon 2.6 GHz with 1 GB memory. The programs were compiled with g++ under Linux using automatic differentiation of the C-XSC library [8] and the interval arithmetic of the Profil/Bias libraries [10]. The results were obtained by running Algorithm MIGO (see Sect. 1) without and with the BTPD method. In order to check the level of improvement of the computational cost, our experiments have been carried out in a wide set of test functions well known in Global Optimization literature.

We present two setups for the BTPD method, which correspond to the two kinds of optimal settings of the parameters α and β. These settings were optimized taking into account the computational effort taken by the algorithms. By computational effort we mean $\#F + n\#G$, where $\#F$ and $\#G$ are the number of inclusion function and inclusion gradient evaluations, respectively. Denoting by E_0 the effort of MIGO algorithm, and by E the effort of MIGO with BTPD, the optimization was done with respect to the total effort (summing up the effort for all the problems) and with respect to the average efficiency rate (the average of E_0/E for all the problems). They will be called setting 1 and 2, respectively. In Tables 1 and 2 the following notation has been used for column headers:

Problem:	The name of the test function.
Ref.:	Reference where the problem is described.
n:	Dimension of the problem.
ε:	Termination criterion ($w(X) \leq \varepsilon$).
E_0:	Computational effort of MIGO.
E_1:	Effort of MIGO with BTPD with $\alpha = 1.45$ and $\beta = 0.028$.
E_2:	Effort of MIGO with BTPD with $\alpha = 0.38$ and $\beta = 0.027$.
T_0:	Computational time in seconds of MIGO.
T_1:	Time of MIGO with BTPD with $\alpha = 1.45$ and $\beta = 0.028$.
T_2:	Time of MIGO with BTPD with $\alpha = 0.38$ and $\beta = 0.027$.

For every problem we have chosen a termination criterion that ensures successful termination within one hour. The data in Table 1 show that the new pruning method reduced the computational effort out of the 42 test problems in 25 and 26 cases for setting 1 and 2, respectively. Table 1 shows that the computational cost is the same (without and with the BTPD method) in 12 and 11 problems for setting 1, 2, respectively, and for 5 problems the effort is greater when the new pruning technique is applied (for both settings). The worst results were obtained for the Rosenbrock-10 function (see Table 2). The reasons can be that the use of the BTPD method changes

Table 1 Computational efficiency comparison

Problem	n	ε	E_0	E_0/E_1	E_0/E_2
Goldstein-Price	2	10^{-10}	33694	1.49	1.54
Six-Hump-Camel-Back	2	10^{-10}	3592	1.26	1.31
Three-Hump-Camel-Back	2	10^{-10}	2328	1.15	1.22
Griewank-2	2	10^{-10}	1238	1.00	1.00
Branin-2	2	10^{-10}	2718	1.01	1.01
Rosenbrock	2	10^{-10}	1319	1.04	1.05
Simplified-Rosenbrock	2	10^{-10}	1319	1.04	1.05
Price	2	10^{-10}	3200	1.00	1.00
Treccani	2	10^{-10}	1506	1.00	1.00
Matyas	2	10^{-10}	2070	1.16	1.16
EX1	2	10^{-10}	468	1.01	0.97
Branin	2	10^{-10}	5527	1.02	1.03
Ratz-4	2	10^{-10}	6210	1.08	1.12
Henriksen-Madsen-3	2	10^{-10}	8631	1.01	1.02
Beale	2	10^{-10}	2755	1.02	1.03
Chichinadze	2	10^{-10}	597	1.00	1.03
Levy-13	2	10^{-10}	281	1.04	1.04
Neumaier3-2	2	10^{-10}	2718	1.29	1.29
Schwefel-2.1	2	10^{-10}	3121	0.97	0.97
Levy-3	2	10^{-10}	5492	1.00	1.00
Levy-5	2	10^{-10}	974	1.00	1.00
Henriksen-Madsen-4	3	10^{-10}	26725	1.02	1.00
Schwefel-3.1	3	10^{-10}	527	1.08	1.08
Hartman-3	3	10^{-10}	4715	1.13	1.18
Levy-8	3	10^{-10}	475	1.03	1.03
Shekel-5	4	10^{-10}	1311	1.00	1.00
Shekel-7	4	10^{-10}	1376	1.00	1.00
Shekel-10	4	10^{-10}	1431	1.00	1.00
Rosenbrock-5	5	10^{-10}	17921	0.88	0.87
Hard-Problem	6	10^{-10}	25	1.00	1.00
Hartman-6	6	10^{-10}	17796	0.97	0.94
Levy-18	7	10^{-10}	1800	1.12	1.12
Levy-12	10	10^{-10}	3111	1.08	1.08
Rosenbrock-10	10	10^{-10}	923896	0.57	0.54
Griewank-10	10	10^{-10}	5361	1.00	1.00
Ratz-5	3	10^{-3}	633508	1.18	1.18
Ratz-6	5	10^{-3}	1201476	1.16	1.16
Ratz-7	7	10^{-3}	1900512	1.14	1.14
Ratz-8	9	10^{-3}	3183190	1.31	1.31
Kowalik	4	10^{-4}	10086339	2.75	2.59
EX2	5	10^{-6}	13851668	2.72	2.82
Rastrigin-10	10	10^{-6}	2319366	0.94	0.94
Average			816007	1.83	1.82
Average of ratios				1.13	1.14

Table 2 Computational time comparison

Problem	n	Ref.	ε	T_0	T_0/T_1	T_0/T_2
Goldstein-Price	2	[18]	10^{-10}	0.31	1.15	1.24
Six-Hump-Camel-Back	2	[18]	10^{-10}	0.03	1.50	1.50
Three-Hump-Camel-Back	2	[18]	10^{-10}	0.01	1.00	1.00
Griewank-2	2	[18]	10^{-10}	0.02	1.00	1.00
Branin-2	2	[18]	10^{-10}	0.05	1.25	1.00
Rosenbrock	2	[4]	10^{-10}	0.01	1.00	1.00
Simplified-Rosenbrock	2	[4]	10^{-10}	0.00	1.00	1.00
Price	2	[5]	10^{-10}	0.02	1.00	1.00
Treccani	2	[4]	10^{-10}	0.00	1.00	1.00
Matyas	2	[19]	10^{-10}	0.00	1.00	1.00
EX1	2	[3]	10^{-10}	0.00	1.00	1.00
Branin	2	[18]	10^{-10}	0.06	1.00	1.00
Ratz-4	2	[17]	10^{-10}	0.10	1.11	1.25
Henriksen-Madsen-3	2	[7]	10^{-10}	0.43	1.00	0.98
Beale	2	[19]	10^{-10}	0.02	1.00	1.00
Chichinadze	2	[5]	10^{-10}	0.01	1.00	1.00
Levy-13	2	[19]	10^{-10}	0.00	1.00	1.00
Neumaier3-2	2	[14]	10^{-10}	0.01	1.00	1.00
Schwefel-2.1	2	[19]	10^{-10}	0.02	0.67	0.67
Levy-3	2	[19]	10^{-10}	0.29	1.00	0.97
Levy-5	2	[19]	10^{-10}	0.05	0.83	1.00
Henriksen-Madsen-4	3	[7]	10^{-10}	1.87	0.97	0.95
Schwefel-3.1	3	[19]	10^{-10}	0.01	1.00	1.00
Hartman-3	3	[18]	10^{-10}	0.08	1.14	1.00
Levy-8	3	[19]	10^{-10}	0.03	1.50	3.00
Shekel-5	4	[18]	10^{-10}	0.03	1.50	1.50
Shekel-7	4	[18]	10^{-10}	0.03	1.00	1.00
Shekel-10	4	[18]	10^{-10}	0.04	1.00	1.00
Rosenbrock-5	5	[14]	10^{-10}	0.16	0.70	0.70
Hard-Problem	6	[17]	10^{-10}	0.00	1.00	1.00
Hartman-6	6	[18]	10^{-10}	0.38	0.97	0.93
Levy-18	7	[19]	10^{-10}	0.08	1.60	1.60
Levy-12	10	[19]	10^{-10}	0.23	1.35	1.21
Rosenbrock-10	10	[14]	10^{-10}	11.94	0.47	0.45
Griewank-10*	10	[18]	10^{-10}	0.20	0.91	0.91
Ratz-5	3	[17]	10^{-3}	125.24	0.94	0.95
Ratz-6	5	[17]	10^{-3}	166.79	0.95	0.95
Ratz-7	7	[17]	10^{-3}	226.56	1.01	0.97
Ratz-8	9	[17]	10^{-3}	582.01	2.18	2.18
Kowalik	4	[19]	10^{-4}	251.29	2.51	2.34
EX2	5	[3]	10^{-6}	279.71	2.18	2.31
Rastrigin-10	10	[14]	10^{-6}	87.74	0.82	0.81
Average				41.33	1.49	1.48
Average of ratios					1.12	1.15

* The searching region of this problem is modified to $[-599, 601]^{10}$

the set of evaluated points that improve the $\tilde{f}$ value, and that the pruning method produces the generation of too many subproblems. For the original Griewank-10 problem we have obtained an extreme speed-up (64.05) when the parameter α was very small, but it was a bit slower when the parameter was higher. The reason is that the minimizer point is in the center of the search region, therefore, when only traditional division is used, the minimizer point is shared by several boxes that cannot be rejected. The asymmetric division generated by the BTPD method avoids this problem, and smaller α provides a greater possibility to do it sooner. This problem was overwhelming the results of the parameter optimization as well, therefore we slightly modified the problem by changing the searching region from $[-600, 600]^{10}$ to $[-599, 601]^{10}$.

The average efficiency rate is 1.13 and 1.14, while the efficiency rate on the total effort is 1.83 and 1.82 for setting 1 and 2, respectively. This clearly shows that on harder problems we can achieve better results, and it can also be seen when focusing on the indicators of the lower part of the table. We can see that the average results obtained by the two settings are almost equal, and the individual results do not differ too much for the two settings either, therefore in general, any value in the same order of magnitude will serve, although will not be the optimal setting for the given problem.

Examining the results of the time comparison we can see that the time ratio is always smaller than the corresponding efficiency rate, due to the time spent on the calculations of the best pruneable box. For instance, the Ratz-5 and Ratz-6 problems need more time with BTPD method (relatively small difference compared to the others) even when the effort is less. Therefore, we can conclude that the application of the new pruning method needs a computational effort that is not negligible but in average, its application for hard to solve problems improves the overall running time.

Future work has to be carried out to tune the variable α accordingly to the problem at hand, or even to every box depending on how promising the actual box is to contain the global minimum. As it was shown in [2], every subproblem needs a different level of division that has to be taken into account to save computations.

6 Summary

This paper investigates the non-suggested possibility to reject areas from multi-dimensional problems based on a generalized construction of a linear lower bounding function using derivative information. The intersection of the linear lower bounding function and the plane of the best upper bound of the minimum gives the rejectable region. The main problem to deal with this is that when interval arithmetic is used to bound the function values in a given multi-dimensional interval, the newly generated sets, by rejecting some area, should be intervals. Therefore, the number of generated subproblems can be great, or the discarded area will be small. This article shows that the use of the Baumann point as a building point for the linear lower bounding function helps to avoid a complicated case analysis and to easily decide when it is worth applying the new pruning/division (BTPD) method and how to do it. The decision to apply or not to apply the new (BTPD) method is based on static parameters that determine the maximum number of generated subproblems and the minimum rejected area, avoiding needle shapes. The presented results show that even with these static settings, the performance of the algorithm is improved, and mostly for hard to solve problems. Future research may contribute to establishing better

and dynamic settings to reduce the computational cost of the presented algorithm even more.

Acknowledgements The authors thank the anonymous referees for their valuable comments and suggestions which improved the quality of this paper significantly.

References

1. Baumann, E.: Optimal centered forms. BIT **28**(1), 80–87 (1988)
2. Casado, L.G., García, I., Csendes, T.: A new multisection technique in interval methods for global optimization. Computing **65**(3), 263–269 (2000)
3. Csendes, T., Ratz, D.: Subdivision direction selection in interval methods for global optimization. SIAM J. Numerical Anal. **34**, 922–938 (1997)
4. Dixon, L., Szego, G. (eds.): Towards Global Optimization. North-Holland Publishing Company, Amsterdam (1975)
5. Dixon, L., Szego, G. (eds.): Towards Global Optimization 2. North-Holland Publishing Company, Amsterdam (1978)
6. Hammer, R., Hocks, M., Kulisch, U., Ratz, D.: C++ Toolbox for Verified Computing I: Basic Numerical Problems: Theory, Algorithms, and Programs. Springer-Verlag, Berlin, Germany (1995)
7. Henriksen, T., Madsen, K.: Use of a depth-first strategy in parallel Global Optimization. Tech. Rep. 92-10, Institute for Numerical Analysis, Technical University of Denmark (1992)
8. Hofschuster, W., Krämer, W.: C-XSC 2.0: A C++ library for extended scientific computing. Numerical software with result verification. Lect. Notes Comput. Sci. **2991**, 15–35 (2004)
9. Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer Academic Publishers, Dordrecht, Holland (1996)
10. Knüppel, O.: PROFIL/BIAS — a fast interval library. Computing **53**, 277–287 (1994)
11. Martínez, J.A., Casado, L.G., García, I., Tóth, B.: Amigo: advanced multidimensional interval analysis global optimization algorithm. In: Floudas, C., Pardalos, P. (eds.) Nonconvex Optimization and Applications Series. Frontiers in Global Optimization, vol. 74, pp. 313–326. Kluwer Academic Publishers (2004)
12. Moore, R.: Interval Analysis. Prentice-Hall, New Jersey, USA (1966)
13. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge (1990)
14. Neumaier, A.: Test functions. World Wide Web, http://www.mat.univie.ac.at/~vpk/math/funcs.html (1999). Used to compare stochastic methods for Global Optimization
15. Ratschek, H., Rokne, J.: New Computer Methods for Global Optimization. Ellis Horwood Ltd., Chichester, England (1988)
16. Ratz, D.: Automatic Slope Computation and its Application in Nonsmooth Global Optimization. Shaker Verlag, Aachen, Germany (1998)
17. Ratz, D., Csendes, T.: On the selection of subdivision directions in interval branch and bound methods for global optimization. J. Global Optimization **7**, 183–207 (1995)
18. Törn, A., Žilinskas, A.: Global Optimization, vol. 3350. Springer-Verlag, Berlin, Germany (1989)
19. Walster, G., Hansen, E., Sengupta, S.: Test results for global optimization algorithm. SIAM Numerical Optimization 1984 pp. 272–287 (1985)