Technische Universität Ilmenau
Institut für Mathematik

Preprint No. M **15/04**

A modification of the alphaBB method for box-constrained optimization and an application to inverse kinematics

Gabriele Eichfelder, Tobias Gerlach and Susanne Sumi

6. Mai 2015

Impressum:
Hrsg.: Leiter des Instituts für Mathematik
Weimarer Straße 25
98693 Ilmenau
Tel.: +49 3677 69-3621
Fax: +49 3677 69-3270
http://www.tu-ilmenau.de/math/

A modification of the αBB method for box-constrained optimization and an application to inverse kinematics

Gabriele Eichfelder[*] , Tobias Gerlach[††] and Susanne Sumi[‡‡]

May 6, 2015

Abstract

For many practical applications it is important to determine not only a numerical approximation of one but a representation of the whole set of globally optimal solutions of a non-convex optimization problem. Then one element of this representation may be chosen based on additional information which cannot be formulated as a mathematical function or within a hierarchical problem formulation. We present such an application in the field of robotic design. This application problem can be modeled as a smooth box-constrained optimization problem. For determining a representation of the global optimal solution set with a predefined quality we modify the well known αBB method. We illustrate the properties and give a proof for the finiteness and correctness of our modified αBB method.

Key Words: Nonconvex programming, global optimization, optimal solution set, αBB method, robotic design

Mathematics subject classifications (MSC 2000): 90C26, 90C30, 90C90

1 Introduction

Many application problems can be modeled as a smooth nonlinear optimization problem with box constraints. This is due to the fact that in technical applications often the design variables are only limited by upper and lower bounds on their range. Numerical solution methods as gradient based methods, sequential quadratic programming or trust region methods evaluate local criteria for optimizing such functions. Therefore, for non-convex problems only locally but not necessarily globally optimal solutions can be guaranteed. However, in applications one is in general only interested in globally optimal solutions. Efficient deterministic solvers for smooth global optimization problems, at least for lower dimensional problems, are available, as for instance the αBB method which is used as the

[*] Institute for Mathematics, Technische Universität Ilmenau, Po 10 05 65, D-98684 Ilmenau, Germany, Gabriele.Eichfelder@tu-ilmenau.de

[††] Institute for Mathematics, Technische Universität Ilmenau, Po 10 05 65, D-98684 Ilmenau, Germany, Tobias.Gerlach@tu-ilmenau.de

[‡‡] Technical Mechanics Group, Technische Universität Ilmenau, Po 10 05 65, D-98684 Ilmenau, Germany, Susanne.Sumi@tu-ilmenau.de

base algorithm in this paper. These algorithms aim in general at the determination of a single globally optimal solution. In contrast, we aim in this paper at finding a representation of the whole optimal solution set with a predefined quality.

The (classical) αBB method is based on results in [5, 17, 18, 19, 20], is described in detail for instance in [1, 2, 9], and is a so-called branch and bound algorithm for determining a single globally optimal solution. The branching is done by a partitioning of the starting box which describes the feasible set. The bounding is reached by minimizing valid convex underestimators of the objective function on each created subbox, and using the minimal value of the convex underestimator as a lower bound. An upper bound can be obtained by evaluating the original objective function at the global minimum point of the convex underestimator.

As a globally optimal solution is in application problems often a possible design, one might be interested not only in one optimal solution but in the whole set of all globally optimal solutions — and thus allowed designs. To be more specific, one is interested in a good representation of the, eventually infinite, optimal solution set, as in practical applications very small changes in the design variables are not practically relevant. We discuss such an application problem in Section 4, which leads to a nonlinear optimization problem with a smooth but non-convex objective function and box constraints. There, the task is to evaluate the design of a robotic arm and to determine whether a desired position can be reached by the robotic arm. All different configurations for the robotic arm such that the position is reached are of interest. This is for instance useful if one wants to move on to another position and has to find the best starting configuration of the robotic arm for doing so. In this application problem it is quite common that many or even infinitely many globally optimal solutions exist.

The topic of this paper is thus a modification of the (classical) αBB method in such a way that it can be used to determine representations of the set of globally optimal solutions with a predefined quality and to apply it to the mentioned application problem. While a naive modification of the αBB method for this purpose seems to be straightforward, we point out arising difficulties and argue while additional variables and special bounds for the additional while loops in the algorithm are necessary. Moreover, we give a proof that our algorithm delivers in a finite number of iterations the desired predefined approximation quality. For this proof we use a box operator, combine several bounds for the objective function on the considered subboxes, and borrow techniques from discrete mathematics.

Another deterministic branch and bound based method with the aim to compute a representation of the whole optimal solution set of a box-constrained optimization problem is the method of Hansen (see [12]). This method computes a representation of all global minimizers of a twice continuously differentiable function in the interior of a given box by using interval arithmetic to evaluate the objective function and its first- and second-order partial derivatives. Based on this the method eliminates by several tests (midpoint test, monotonicity test, concavity test) subboxes, which are guaranteed not to contain a global minimizer. However, this method does not find non-stationary minima on the boundary of the feasible set (for an amendment in this regard see [28]) and cannot be transferred to general constrained optimization problems directly. A detailed description of this method and its modifications can be found in [11, 13] and the references therein.

The remaining of the paper is organized as follows. In section 2 we present some basic notations and definitions. We collect some properties of a convex underestimator, prove some first results, and introduce a box operator used in the branching part of the algorithms. Our modified αBB method is formulated in section 3. We give a proof for the finiteness and correctness of this method, we describe possible variations, and we present first numerical results. In section 4 we apply our method to a design problem in the scope of robotics. Finally, section 5 concludes the paper and gives some directions of future research.

2 Preliminaries

We denote by $\mathbb{IR}$ the set of all real nonempty closed intervals and by $\mathbb{IR}^n$ the set of all n-dimensional boxes. For some $X = (X_1, \ldots, X_n)^\top \in \mathbb{IR}^n$ we define the vectors $l(X) = (l(X)_1, \ldots, l(X)_n)^\top$, $u(X) = (u(X)_1, \ldots, u(X)_n)^\top$, and $\mathrm{mid}(X) = (\mathrm{mid}(X)_1, \ldots, \mathrm{mid}(X)_n)^\top$ by

$$l(X)_i := \min(X_i),\ u(X)_i := \max(X_i)\ \text{, and }\ \mathrm{mid}(X)_i = \frac{1}{2}\left(u(X)_i + l(X)_i\right)$$

for all $i \in \{1, \ldots, n\}$, respectively. For a given box $X \in \mathbb{IR}^n$ we will use the notation $X = [l(X), u(X)]$ simultaneously and define the width of X by $\omega(X) := \|u(X) - l(X)\|_2$.

For reasons of simplicity we consider throughout the paper a box-constrained optimization problem

$$\min_{x \in X^0}\ f(x) \tag{1}$$

for some $X^0 \in \mathbb{IR}^n$ and some $f \in C^2(\mathbb{R}^n, \mathbb{R})$, where $C^2(\mathbb{R}^n, \mathbb{R})$ denotes the set of all twice continuously differentiable functions on $\mathbb{R}^n$. At the end of this paper we shortly comment on how general constraints can be incorporated, see also [9, Chapter 12].

For the representation of the set of minimal solutions of the optimization problem (1) we will use the following definitions:

Definition 2.1. *Let $\varepsilon > 0$, $\delta > 0$, Ω be a nonempty subset of $\mathbb{R}^n$, and $f : \Omega \to \mathbb{R}$ such that* $\operatorname{argmin}_{x \in \Omega} f(x) \neq \emptyset$.

(a) A point $\tilde{x} \in \Omega$ is an ε-minimal point *of f w.r.t. Ω, if*

$$f(\tilde{x}) - \min_{x \in \Omega} f(x) \leq \varepsilon.$$

(b) A point $\hat{x} \in \Omega$ is an (ε, δ)-minimal point *of f w.r.t. $\bar{x} \in \operatorname{argmin}_{x \in \Omega} f(x)$, if*

$$f(\hat{x}) - \min_{x \in \Omega} f(x) \leq \varepsilon \textit{ and } \|\hat{x} - \bar{x}\|_2 \leq \delta.$$

(c) A finite subset A of Ω is an (ε, δ)-minimal set *of f w.r.t. Ω if every point $\tilde{x} \in A$ is an ε-minimal point of f w.r.t. Ω and if for every $\bar{x} \in \operatorname{argmin}_{x \in \Omega} f(x)$ there exists a point $\hat{x} \in A$ such that $\hat{x}$ is an (ε, δ)-minimal point of f w.r.t. $\bar{x}$.*

Based on the above definition we denote by $\varepsilon\text{-}\min(f, \Omega)$ and by $(\varepsilon, \delta)\text{-}\min(f, \Omega, \bar{x})$ the set of all ε-minimal points of f w.r.t. Ω and the set of all (ε, δ)-minimal points of f w.r.t. $\bar{x} \in \operatorname{argmin}_{x \in \Omega} f(x)$, respectively.

For constructing lower bounds of the minimum value of (1) in the bounding step of algorithms based on the αBB approach (valid) convex underestimators are of central importance. Referring for instance to [2], an essential step for the construction of a valid convex underestimator is the decomposition of the objective function into a sum of linear, bilinear, trilinear, fractional, fractional trilinear, convex, univariate concave, and general non-convex terms. All of the linear and all of the convex terms do not require any transformations for the construction of such an underestimator, and all univariate concave terms can be underestimated separately by linear functions without the introduction of additional variables or constraints (see [2]). For bilinear, trilinear, fractional, and fractional trilinear terms there exist sophisticated techniques which generate valid and in some cases very tight convex underestimators (see [6, 21, 22]). We restrict ourselves in this paper to the case of general non-convex terms and use the classical approaches of [5] and [20]. Here, the convex underestimator is constructed by adding a quadratic term to the objective function built by a lower bound of the smallest eigenvalue of the Hessian of the objective function over the subboxes derived by interval analysis. For improved approaches for the construction of convex underestimators we refer for instance to [3, 4].

Definition 2.2. *Let Ω be a nonempty convex subset of $\mathbb{R}^n$ and $f : \Omega \to \mathbb{R}$. A function $\Phi : \tilde{\Omega} \to \mathbb{R}$ with $\Omega \subset \tilde{\Omega}$ is a* convex underestimator *of f w.r.t. Ω, if Φ is convex on Ω and $\Phi(x) \leq f(x)$ for all $x \in \Omega$.*

For our theoretical results we restrict ourselves with regard to the construction of the convex underestimators exemplary, as mentioned above, to the classical approach of [20]. Therefor the twice continuously differentiable objective function f will be underestimated on the box $X = [l(X), u(X)] \in \mathbb{R}^n$ by a function $\Phi_{\alpha,X} : \mathbb{R}^n \to \mathbb{R}$ defined by

$$\Phi_{\alpha,X}(x) := f(x) + \alpha \sum_{i=1}^{n} (l(X)_i - x_i)(u(X)_i - x_i) \tag{2}$$

with $\alpha \geq 0$. We denote by $\nabla f(x)$ the gradient of f and by $\nabla^2 f(x)$ the Hessian of f at the point $x \in \mathbb{R}^n$. Moreover, we denote by $\lambda_{\min}(A)$ the smallest eigenvalue of a symmetric matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$.

The following lemma is obvious:

Lemma 2.3. *Let $X^0 \in \mathbb{R}^n$, $f \in C^2(\mathbb{R}^n, \mathbb{R})$, $X \in \mathbb{R}^n$ such that $X \subset X^0$ and*

$$X \cap \operatorname*{argmin}_{x \in X^0} f(x) \neq \emptyset,$$

$\alpha \geq 0$, and $\Phi_{\alpha,X} : X \to \mathbb{R}$ be defined as in (2). Then it holds

$$\min_{x \in X} \Phi_{\alpha,X}(x) \leq \min_{x \in X^0} f(x).$$

The following two results are well known:

Lemma 2.4. *[20, Property 3]*
Let $X \in \mathbb{R}^n$, $f \in C^2(\mathbb{R}^n, \mathbb{R})$, and $\Phi_{\alpha,X} : \mathbb{R}^n \to \mathbb{R}$ be defined as in (2). Then the function $\Phi_{\alpha,X}$ is a convex underestimator of f w.r.t. X if and only if

$$\alpha \geq \max\left\{0, \; -\frac{1}{2} \min\left\{ \lambda_{\min}(\nabla^2 f(x)) \middle| \; x \in X \right\}\right\}. \tag{3}$$

Lemma 2.5. *[20, Property 4]*
Let $X \in \mathbb{R}^n$, $f \in C^2(\mathbb{R}^n, \mathbb{R})$, and $\Phi_{\alpha,X} : \mathbb{R}^n \to \mathbb{R}$ be defined as in (2)*. Then it holds*

$$\max_{x \in X}(f(x) - \Phi_{\alpha,X}(x)) = \frac{1}{4}\alpha\ \omega^2(X). \tag{4}$$

Note, if the function $f \in C^2(\mathbb{R}^n, \mathbb{R})$ is convex on X, then $\lambda_{\min}(\nabla^2 f(x)) \geq 0$ holds for all $x \in X$ and $\alpha = 0$ satisfies (3). Moreover, (4) delivers a (natural) measure for the quality of the underestimator $\Phi_{\alpha,X}$.

Lemma 2.6.
Let $X^1, X^2 \in \mathbb{R}^n$ with $X^2 \subset X^1$, $f \in C^2(\mathbb{R}^n, \mathbb{R})$, and $\Phi_{\alpha,X^k} : \mathbb{R}^n \to \mathbb{R}$ be defined as in (2) *for $k \in \{1, 2\}$. If Φ_{α,X^1} is a convex underestimator of f w.r.t. X^1, then Φ_{α,X^2} is also a convex underestimator of f w.r.t. X^2 and it holds*

$$\min_{x \in X^1} \Phi_{\alpha,X^1}(x) \leq \min_{x \in X^2} \Phi_{\alpha,X^2}(x).$$

Proof. Let Φ_{α,X^1} be a convex underestimator of f w.r.t. X^1 and thus also w.r.t. $X^2 \subset X^1$. Using

$$\min_{x \in X^1} \lambda_{\min}(\nabla^2 f(x)) \leq \min_{x \in X^2} \lambda_{\min}(\nabla^2 f(x))$$

the first conclusion follows immediately by Lemma 2.4. By using $X^2 \subset X^1$ it follows $l(X^1)_i \leq l(X^2)_i < u(X^2)_i \leq u(X^1)_i$ for all $i \in \{1, \ldots, n\}$. Hence, for all $x \in X^2$ we obtain

$$\begin{aligned}
\Phi_{\alpha,X^1}(x) &= f(x) + \alpha \sum_{i=1}^{n}(l(X^1)_i - x_i)(u(X^1)_i - x_i) \\
&\leq f(x) + \alpha \sum_{i=1}^{n}(l(X^2)_i - x_i)(u(X^2)_i - x_i) \\
&= \Phi_{\alpha,X^2}(x)
\end{aligned}$$

and therefore

$$\min_{x \in X^1} \Phi_{\alpha,X^1}(x) \leq \min_{x \in X^2} \Phi_{\alpha,X^1}(x) \leq \min_{x \in X^2} \Phi_{\alpha,X^2}(x).$$

$\square$

For a given box $X = [l(X), u(X)] \in \mathbb{R}^n$ we define the branching index $\mathrm{b}(X)$ by

$$\mathrm{b}(X) := \min\left\{ i \in \{1, \ldots, n\} \;\middle|\; i \in \operatorname*{argmax}_{j=1,\ldots,n}(u(X)_j - l(X)_j) \right\} \tag{5}$$

and the subboxes $L(X)$ and $R(X)$ of X by

$$\begin{array}{lll}
l(L(X))_i := l(X)_i, & u(L(X))_{\mathrm{b}(X)} := \mathrm{mid}(X)_{\mathrm{b}(X)}, & u(L(X))_j := u(X)_j, \\
l(R(X))_j := l(X)_j, & l(R(X))_{\mathrm{b}(X)} := \mathrm{mid}(X)_{\mathrm{b}(X)}, & u(R(X))_i := u(X)_i
\end{array} \tag{6}$$

for all $i \in \{1, \ldots, n\}$ and all $j \in \{1, \ldots, n\} \setminus \{\mathrm{b}(X)\}$, respectively. According to this we define the box operator $\mathrm{sB} : \mathbb{R}^n \times \mathbb{N}_0 \to 2^X$ recursively by

$$\begin{array}{ll}
\mathrm{sB}(X, 0) := \{X\} & \text{for all } X \in \mathbb{R}^n, \\
\mathrm{sB}(X, 1) := \{L(X), R(X)\} & \text{for all } X \in \mathbb{R}^n, \\
\mathrm{sB}(X, j) := \mathrm{sB}(L(X), j-1) \cup \mathrm{sB}(R(X), j-1) & \text{for all } X \in \mathbb{R}^n \text{ and all } j \geq 2.
\end{array}$$

Obviously for all $X \in \mathbb{IR}^n$ and all $j \in \mathbb{N}_0$ it holds

$$|\,\mathrm{sB}(X,j)| = 2^j \quad \text{and} \quad |\bigcup_{j=0}^{m} \mathrm{sB}(X,j)| = 2^{m+1} - 1. \tag{7}$$

For the width of a subbox $\tilde{X}$ of X defined by the box operator sB we obtain:

Lemma 2.7. *Let $X \in \mathbb{IR}^n$ and $j \in \mathbb{N}_0$. Then it holds*

$$\omega(\tilde{X}) \leq \omega(X) \left(1 - \frac{3}{4n}\right)^{\frac{j}{2}}$$

for all $\tilde{X} \in \mathrm{sB}(X,j)$.

Proof.
For $j = 0$ the assertion is obvious. For $j = 1$ and $\tilde{X} \in \mathrm{sB}(X,1)$ it follows by (5) and (6)

$$\omega^2(X) = \sum_{i=1}^{n} \left(u\left(X\right)_i - l\left(X\right)_i\right)^2 \leq n \left(u\left(X\right)_{\mathrm{b}(X)} - l\left(X\right)_{\mathrm{b}(X)}\right)^2$$

and

$$\omega^2(\tilde{X}) = \sum_{i=1}^{n} \left(u(\tilde{X})_i - l(\tilde{X})_i\right)^2 = \omega^2\left(X\right) - \frac{3}{4}\left(u\left(X\right)_{\mathrm{b}(X)} - l\left(X\right)_{\mathrm{b}(X)}\right)^2.$$

Hence, we obtain $\frac{\omega^2(\tilde{X})}{\omega^2(X)} \leq 1 - \frac{3}{4n}$ and the assertion follows by mathematical induction on j. □

Remark 2.8. *Clearly it holds $\lim\limits_{n\to\infty}\left(1 - \frac{3}{4n}\right) = 1$, which illustrates the deteriorating convergence properties for branch and bound algorithms used in the case of high dimensional optimization problems.*

3 The modified αBB method

We formulate the (basic version) of the *modified αBB method* for the computation of an (ε,δ)-minimal set A of f w.r.t. X^0 for the optimization problem (1) as follows:

Algorithm 1 The modified αBB method

```
INPUT: $X^0 \in \mathbb{R}^n$, $f \in C^2(\mathbb{R}^n, \mathbb{R})$, $\varepsilon > 0$, $\delta > 0$
OUTPUT: $(\varepsilon, \delta)$-minimal set $A$ of $f$ w.r.t. $X^0$
 1: Compute an $\alpha \geq 0$ such that $\Phi_{\alpha,X^0}$ is a convex underestimator of $f$ w.r.t. $X^0$.
 2: Set $A := \emptyset$, $X^\star := X^0$, $x^\star := \mathrm{mid}(X^\star)$, $u^\star := -\infty$, $x_{\mathrm{act}} := x^\star$, $v_{\mathrm{act}} := +\infty$, $v_{\mathrm{glob}} := v_{\mathrm{act}}$,
    $\mathcal{L} := \{(X^\star, x^\star, u^\star)\}$, and the iteration counter $k := 0$.
 3: while $\mathcal{L} \neq \emptyset$ do
 4:     Set $k := k + 1$.
 5:     Delete $(X^\star, x^\star, u^\star)$ from $\mathcal{L}$.
 6:     for all $\bar{X} \in \mathrm{sB}(X^\star, 1)$ do
 7:         Compute $\bar{x} \in \mathrm{argmin}_{x \in \bar{X}} \Phi_{\alpha,\bar{X}}(x)$ and $\bar{u} := \min_{x \in \bar{X}} \Phi_{\alpha,\bar{X}}(x)$.
 8:         if $\bar{u} \leq v_{\mathrm{glob}}$ then
 9:             Add $(\bar{X}, \bar{x}, \bar{u})$ as the last element to $\mathcal{L}$.
10:             if $f(\bar{x}) \leq v_{\mathrm{act}}$ then
11:                 Set $x_{\mathrm{act}} := \bar{x}$, $v_{\mathrm{act}} := f(x_{\mathrm{act}})$, and $v_{\mathrm{glob}} := \min\{v_{\mathrm{act}},\ v_{\mathrm{glob}}\}$.
12:                 Delete all $(\tilde{X}, \tilde{x}, \tilde{u}) \in \mathcal{L}$ with $\tilde{u} > v_{\mathrm{glob}}$ from $\mathcal{L}$.
13:             end if
14:         end if
15:     end for
16:     if $\mathcal{L} \neq \emptyset$ then
17:         Define $(X^\star, x^\star, u^\star)$ as the first element of $\mathcal{L}$ with $u^\star = \min_{(\tilde{X},\tilde{x},\tilde{u}) \in \mathcal{L}} \tilde{u}$.
18:         while $\mathcal{L} \neq \emptyset$ and $v_{\mathrm{act}} - u^\star \leq \frac{1}{2}\varepsilon$ do
19:             if there is a $(\hat{X}, \hat{x}, \hat{u}) \in \mathcal{L}$ with $x_{\mathrm{act}} \in \hat{X}$ and $\omega(\hat{X}) \leq \delta$ then
20:                 Set $A := A \cup \{x_{\mathrm{act}}\}$.
21:                 Delete all $(\tilde{X}, \tilde{x}, \tilde{u}) \in \mathcal{L}$ with $x_{\mathrm{act}} \in \tilde{X}$ and $\omega(\tilde{X}) \leq \delta$ from $\mathcal{L}$.
22:             end if
23:             if there is a $(\hat{X}, \hat{x}, \hat{u}) \in \mathcal{L}$ with $x_{\mathrm{act}} \in \hat{X}$ then
24:                 Define $(X^\star, x^\star, u^\star)$ as the first element of $\mathcal{L}$ with $x_{\mathrm{act}} \in X^*$.
25:                 go to 3.
26:             end if
27:             if $\mathcal{L} \neq \emptyset$ then
28:                 Define $(X^\star, x^\star, u^\star)$ as the first element of $\mathcal{L}$ with $u^\star = \min_{(\tilde{X},\tilde{x},\tilde{u}) \in \mathcal{L}} \tilde{u}$.
29:                 Compute the first element $(\cdot, \dot{x}, \cdot) \in \mathrm{argmin}_{(\tilde{X},\tilde{x},\tilde{u}) \in \mathcal{L}} f(\tilde{x})$ of $\mathcal{L}$.
30:                 Set $x_{\mathrm{act}} := \dot{x}$ and $v_{\mathrm{act}} := f(x_{\mathrm{act}})$.
31:             end if
32:         end while
33:     end if
34: end while
```

The modified αBB method starts with the construction of a convex underestimator Φ_{α,X^0} of the objective function f w.r.t. the starting box X^0 (line 1) and with the initialization of the algorithm (line 2). The outer while loop (lines 3-34) includes the iteration counter (line 4), a branching and bounding part (line 5 and the for-loop of lines 6-15), and the selection part for the elements of the (ε, δ)-minimal set A of f w.r.t. X^0 (lines 16-33). Obviously, the algorithm terminates in an iteration $k \geq 1$ if $\mathcal{L} = \emptyset$ holds in line 3.

In every iteration $k \geq 1$ the current element $(X^\star, x^\star, u^\star)$ is deleted from $\mathcal{L}$ in line 5 and the corresponding box $X^\star$ is subdivided by the box operator sB in two subboxes. For a subbox $\bar{X} \in \mathrm{sB}(X^\star, 1) = \{L(X^\star), R(X^\star)\}$ any local minimum of the corresponding convex underestimator $\Phi_{\alpha,\bar{X}}$ over the convex set $\bar{X}$ is also a global minimum. Thus in the following bounding part (lines 6-15) local optimization techniques (for instance SQP methods) can be used to determine a point $\bar{x} \in \operatorname{argmin}_{x \in \bar{X}} \Phi_{\alpha,\bar{X}}(x)$. Then, we can use

$$\bar{u} = \min_{x \in \bar{X}} \Phi_{\alpha,\bar{X}}(x) \leq \min_{x \in \bar{X}} f(x) \tag{8}$$

as a lower bound of f over $\bar{X}$, cf. line 7. Obviously it holds

$$\min_{x \in X^0} f(x) \leq v_{\mathrm{glob}} \leq v_{\mathrm{act}} = f(x_{\mathrm{act}}) \tag{9}$$

in every iteration k of Algorithm 1, where v_{glob} denotes the current upper bound of $\min_{x \in X^0} f(x)$.

If $\bar{u} > v_{\mathrm{glob}}$ is fulfilled in line 8, then it follows by (8) and (9)

$$\min_{x \in X^0} f(x) \leq v_{\mathrm{glob}} < \bar{u} \leq \min_{x \in \bar{X}} f(x).$$

Hence, it holds $\bar{X} \cap \ \operatorname{argmin}_{x \in X^0} f(x) = \emptyset$ and the corresponding element $(\bar{X}, \bar{x}, \bar{u})$ will not be added to $\mathcal{L}$.

If $\bar{u} \leq v_{\mathrm{glob}}$ is fulfilled in line 8, then $\bar{X}$ can contain an element of $\operatorname{argmin}_{x \in X^0} f(x)$ and the corresponding element $(\bar{X}, \bar{x}, \bar{u})$ will be added to $\mathcal{L}$ in line 9. In this case the algorithm tries in the following if-loop (lines 10-13) to improve the values of $v_{\mathrm{act}} = f(x_{\mathrm{act}})$ and the current upper bound v_{glob} by using $f(\bar{x})$. Where applicable, all elements $(\tilde{X}, \tilde{x}, \tilde{u}) \in \mathcal{L}$ with $\tilde{X} \cap \ \operatorname{argmin}_{x \in X^0} f(x) = \emptyset$ (guaranteed by the improved upper bound v_{glob}) are deleted (bounding).

To guarantee uniqueness in the choice of the subbox $\bar{X}$ we determine that the algorithm analyzes in the bounding part of every iteration $k \geq 1$ at first $L(X^\star)$ and then $R(X^\star)$.

In the selection part (lines 16-33) the point x_{act} and the value v_{act} are of central importance. We waive a detailed description of this part and we remark only that $x_{\mathrm{act}} \in \varepsilon\text{-}\min(f, X^0)$ holds if $\mathcal{L} \neq \emptyset$ and $v_{\mathrm{act}} - u^\star \leq \frac{1}{2}\varepsilon$ are fulfilled in line 18 (see the forthcoming Lemma 3.4). Note that this conclusion does not hold in general for the weaker condition $v_{\mathrm{act}} - u^\star \leq \varepsilon$ as the following example illustrates:

Example 3.1. *Note that all results of this example have to be interpreted in view of the numerical accurateness of* Matlab. *We consider the box-constrained optimization problem* (1) *with $f : \mathbb{R} \to \mathbb{R}$, $f(x) := -10^{-6}\,(\sin(x + 10.5))^2\,(x+10.5)^6$, $X^0 := [0, 4]$, and the unique global minimum x_{opt} of f over X^0 at*

$$x_{\mathrm{opt}} = 3.84335076139211 \quad \textit{with} \quad f(x_{\mathrm{opt}}) = -8.34274122196571.$$

For $\alpha := 6$ the function $\Phi_{6,[0,4]} : \mathbb{R} \to \mathbb{R}$ defined by

$$\Phi_{6,[0,4]}(x) := -10^{-6}\,(\sin(x + 10.5))^2\,(x + 10.5)^6 + 6x(x - 4)$$

is a convex underestimator of f w.r.t. X^0.

Using Algorithm 1 with $\varepsilon := 6$ and $\delta := 3$ we obtain after two iterations

$$A = \{3.70082023715804\}$$

with

$$f(3.70082023715804) = -8.16807544598965$$

which is in fact an (ε, δ)-minimal set of f w.r.t. X^0.

If we modify the selection part by using the weaker condition $v_{\text{act}} - u^\star \leq \varepsilon$, then the modified algorithm delivers after one iteration

$$A = \{3.46575415683705,\ 0.93618642367221\}$$

with

$$f(3.46575415683705) = -7.20381885980831,$$
$$f(0.93618642367221) = -1.83018768013290,$$

and this set A is obviously not an (ε, δ)-minimal set of f w.r.t. X^0.

3.1 Finiteness and correctness

For the proof of the main result of this subsection (see the forthcoming Theorem 3.6) we need some preliminary conclusions, which we collect at first. Thereby the following remark follows immediately by Lemma 2.3:

Remark 3.2.

(i) If in iteration k of Algorithm 1 an element $(X^\star, x^\star, u^\star)$ with

$$X^\star \cap \operatorname*{argmin}_{x \in X^0} f(x) \neq \emptyset$$

is deleted from $\mathcal{L}$ in line 5, then for at least one $\bar{X} \in \mathrm{sB}(X^\star, 1)$ it holds

$$\bar{X} \cap \operatorname*{argmin}_{x \in X^0} f(x) \neq \emptyset$$

and an element $(\bar{X}, \bar{x}, \bar{u})$ with $\bar{x} \in \operatorname{argmin}_{x \in \bar{X}} \Phi_{\alpha,\bar{X}}(x)$ and $\bar{u} := \min_{x \in \bar{X}} \Phi_{\alpha,\bar{X}}(x)$ is added to $\mathcal{L}$ in iteration k in line 9.

(ii) There exists no iteration k of Algorithm 1 such that an element $(\tilde{X}, \tilde{x}, \tilde{u})$ with

$$\tilde{X} \cap \operatorname*{argmin}_{x \in X^0} f(x) \neq \emptyset$$

is deleted from $\mathcal{L}$ in line 12.

Lemma 3.3. *Let k_0 be the first iteration of Algorithm 1 such that $\mathcal{L} \neq \emptyset$ and $v_{\text{act}} - u^\star \leq \frac{1}{2}\varepsilon$ are fulfilled in line 18. Then for all iteration $k \geq k_0$ it holds*

$$v_{\text{glob}} - \min_{x \in X^0} f(x) \leq \frac{1}{2}\varepsilon.$$

Proof. By the choice of k_0 it follows by Remark 3.2(*i*) and (*ii*) that in iteration k_0 there exists an element $(X, x, u) \in \mathcal{L}$ with $X \cap \ \operatorname{argmin}_{x \in X^0} f(x) \neq \emptyset$. Moreover, by the choice of $u^\star$ (see line 17) it follows by Lemma 2.3 that $u^\star \leq u \leq \min_{x \in X^0} f(x)$. Hence, we obtain by (9)

$$v_{\text{glob}} - \min_{x \in X^0} f(x) \leq v_{\text{act}} - \min_{x \in X^0} f(x) \leq v_{\text{act}} - u^\star \leq \frac{1}{2}\varepsilon.$$

Since the value of v_{glob} does not increase in all iterations $k \geq k_0$, we are done. □

Lemma 3.4. *Let k_0 be defined as in Lemma 3.3. Then for all iteration $k \geq k_0$ of Algorithm 1, with $\mathcal{L} \neq \emptyset$ and $v_{\text{act}} - u^\star \leq \frac{1}{2}\varepsilon$ being fulfilled in line 18, it holds*

$$x_{\text{act}} \in \varepsilon\text{-}\min(f, X^0).$$

Proof. By Lemma 3.3 it holds $v_{\text{glob}} - \frac{1}{2}\varepsilon \leq \min_{x \in X^0} f(x)$ for all $k \geq k_0$. Hence, by using $v_{\text{act}} \leq u^\star + \frac{1}{2}\varepsilon$ we obtain

$$v_{\text{act}} - \min_{x \in X^0} f(x) \leq v_{\text{act}} - v_{\text{glob}} + \frac{1}{2}\varepsilon \leq u^\star - v_{\text{glob}} + \varepsilon.$$

Since $\tilde{u} - v_{\text{glob}} \leq 0$ holds for all $(\tilde{X}, \tilde{x}, \tilde{u}) \in \mathcal{L}$ (see lines 8 - 14) it follows

$$f(x_{\text{act}}) = v_{\text{act}} \leq \min_{x \in X^0} f(x) + u^\star - v_{\text{glob}} + \varepsilon \leq \min_{x \in X^0} f(x) + \varepsilon,$$

and we are done. □

In the following $\lceil \cdot \rceil$ denotes the ceiling function, i.e.

$$\lceil x \rceil := \min \{n \in \mathbb{N} \mid n \geq x\} \text{ for all } x \in \mathbb{R}.$$

Lemma 3.5. *Let Algorithm 1 attain in iteration k line 16, let $(\hat{X}, \hat{x}, \hat{u})$ be an element of $\mathcal{L}$, and let $j \in \mathbb{N}_0$ such that $\hat{X} \in \mathrm{sB}(X^0, j)$.*

(*i*) *If*

$$j \geq j_\delta := \left\lceil 2 \, \frac{\log\left(\frac{\delta}{\omega(X^0)}\right)}{\log\left(1 - \frac{3}{4n}\right)} \right\rceil, \tag{10}$$

then it holds $\omega(\hat{X}) \leq \delta$.

(*ii*) *If*

$$j \geq j_\varepsilon := \left\lceil 2 \, \frac{\log\left(\frac{\sqrt{\frac{2\varepsilon}{\alpha}}}{\omega(X^0)}\right)}{\log\left(1 - \frac{3}{4n}\right)} \right\rceil, \tag{11}$$

then it holds $v_{\text{act}} - \hat{u} \leq \frac{1}{2}\varepsilon$.

Proof.
(*i*) Follows by Lemma 2.7 since

$$j \geq 2 \, \frac{\log\left(\frac{\delta}{\omega(X)}\right)}{\log\left(1 - \frac{3}{4n}\right)} \quad \Leftrightarrow \quad \omega(X) \left(1 - \frac{3}{4n}\right)^{\frac{j}{2}} \leq \delta.$$

(ii) It is easy to see that regardless whether the current value of v_{act} in iteration k was defined in an iteration $k^{'} \leq k$ in line 11 or in line 30 it holds $v_{\text{act}} \leq f(\hat{x})$ (see line 10 or line 29, respectively). Using Lemma 2.5 and *(i)* we obtain

$$v_{\text{act}} - \hat{u} \leq f(\hat{x}) - \hat{u} \leq \frac{1}{4}\alpha\omega^2(\hat{X}) \leq \frac{1}{4}\alpha\frac{2\varepsilon}{\alpha} = \frac{1}{2}\varepsilon.$$

□

Let k_0 be defined as in Lemma 3.3 and j_ε be defined as in (11). By using Lemma 3.5*(ii)* we can formulate an upper bound for k_0. In every iteration $k \geq 1$ of Algorithm 1 where we reach line 16 with $\mathcal{L} \neq \emptyset$ then in the next step, in line 17, an element $(X^\star, x^\star, u^\star) \in \mathcal{L}$ is chosen as the first element of $\mathcal{L}$ with $u^\star = \min_{(\tilde{X},\tilde{x},\tilde{u})\in\mathcal{L}} \tilde{u}$. If $v_{\text{act}} - u^\star > \frac{1}{2}\varepsilon$, then by Lemma 3.5*(ii)* it follows $X^\star \in \text{sB}(X^0, j)$ for some j with $j \leq j_\varepsilon - 1$. If $k < k_0$, then in iteration $k+1$ the element $(X^\star, x^\star, u^\star)$ will be deleted in line 5 and the box $X^\star$ will be subdivided in line 6-15. Hence, in all iterations $k < k_0$ every element $(X^\star, x^\star, u^\star) \in \mathcal{L}$ can only be chosen once in line 17. The question arises, how many elements $(X^\star, x^\star, u^\star) \in \mathcal{L}$ can be chosen in line 17 with $X^\star \in \text{sB}(X^0, j)$ in the worst case such that $j \leq j_\varepsilon - 1$ holds, before the algorithm must chose an element of $\mathcal{L}$ with $X^\star \in \text{sB}(X^0, j)$ and $j = j_\varepsilon$.

Therefore, since it holds by (7)

$$|\bigcup_{j=0}^{j_\varepsilon - 1} \text{sB}(X^0, j)| = 2^{j_\varepsilon} - 1,$$

there must be an iteration $k_0 \leq 2^{j_\varepsilon}$ such that $\mathcal{L} \neq \emptyset$ holds in line 16, an element $(X^\star, x^\star, u^\star) \in \mathcal{L}$ is chosen in line 17 with $X^\star \in \text{sB}(X^0, j_\varepsilon)$, and $v_{\text{act}} - u^\star \leq \frac{1}{2}\varepsilon$ is fulfilled in line 18 by Lemma 3.5*(ii)*.

Let $\mathcal{X}(0) := \{X^0\}$. If Algorithm 1 attains in iteration $k \geq 1$ line 5, then an element $(X^\star, x^\star, u^\star)$ is deleted from $\mathcal{L}$ and we define recursively for $k \geq 1$

$$\mathcal{X}(k) := \mathcal{X}(k-1) \cup \text{sB}(X^\star, 1),$$

i.e. $\mathcal{X}(k)$ contains X^0 and all subboxes of X^0 constructed by Algorithm 1 up to and including iteration $k \geq 1$. Obviously it holds

$$|\mathcal{X}(k)| = 2k + 1 \text{ for all } k \geq 0. \tag{12}$$

Using the above considerations we are now able to formulate and prove our main result:

Theorem 3.6. *For Algorithm 1 the following holds:*

(i) The algorithm terminates after finitely many iterations.

(ii) At the end of the algorithm the set A is an (ε, δ)-minimal set of f w.r.t. X^0.

Proof.
(i) Let j_δ be defined as in (10) and j_ε be defined as in (11), respectively. If Algorithm 1 attains in iteration k line 16 and $(\hat{X}, \hat{x}, \hat{u})$ is an element of $\mathcal{L}$ with $\hat{X} \in \text{sB}(X^0, j_{\delta,\varepsilon})$ and

$$j_{\delta,\varepsilon} := \max\{j_\delta, j_\varepsilon\},$$

then by Lemma 3.5(i) and (ii) it holds $\omega(\hat{X}) \leq \delta$ and $v_{\mathrm{act}} - \hat{u} \leq \frac{1}{2}\varepsilon$. Hence, $(\hat{X}, \hat{x}, \hat{u})$ cannot be chosen as $(X^\star, x^\star, u^\star)$ in an iteration $k' \geq k$ in line 24. Moreover, if $(\hat{X}, \hat{x}, \hat{u})$ will be chosen as $(X^\star, x^\star, u^\star)$ in an iteration $k' \geq k$ in line 17 or in line 28, respectively, then the Algorithm 1 will stay in the while-loop between line 18 and line 32 with $(X^\star, x^\star, u^\star) = (\hat{X}, \hat{x}, \hat{u})$ until $(\hat{X}, \hat{x}, \hat{u})$ is deleted from $\mathcal{L}$. Hence, there exists no iteration $k' \geq k$ such that $(X^\star, x^\star, u^\star) = (\hat{X}, \hat{x}, \hat{u})$ will be deleted from $\mathcal{L}$ in line 5 and such that $X^\star = \hat{X}$ will be subdivided in line 6-15. Thus $\mathcal{X}(k) \cap \mathrm{sB}(X^0, j_{\delta,\varepsilon} + 1) = \emptyset$ holds for all iterations $k \geq 1$. Since by (7) and (12) it holds

$$|\bigcup_{j=0}^{j_{\delta,\varepsilon}} \mathrm{sB}(X^0, j)| = |\mathcal{X}(\bar{k})| \Leftrightarrow 2^{j_{\delta,\varepsilon}+1} - 1 = 2\bar{k} + 1 \Leftrightarrow \bar{k} = 2^{j_{\delta,\varepsilon}} - 1,$$

the Algorithm 1 terminates in an iteration $k \leq 2^{j_{\delta,\varepsilon}} - 1$.
(ii) Using Lemma 3.4 it remains to show that at the end of Algorithm 1 for all $\bar{x} \in \operatorname{argmin}_{x\in\Omega} f(x)$ there exists an $\hat{x} \in A$ such that $\|\hat{x} - \bar{x}\|_2 \leq \delta$. Assume, there is at the end of the algorithm a point $\bar{x} \in \operatorname{argmin}_{x\in\Omega} f(x)$ such that

$$\|\hat{x} - \bar{x}\|_2 > \delta \tag{13}$$

holds for all $\hat{x} \in A$. In addition to that let $\bar{k}$ be the last iteration where an element $(\breve{X}, \breve{x}, \breve{u})$ with $\bar{x} \in \breve{X}$ is deleted from $\mathcal{L}$. By Remark 3.2 (i) and (ii) the element $(\breve{X}, \breve{x}, \breve{u})$ cannot be deleted in line 5 or in line 12, respectively. Hence, $(\breve{X}, \breve{x}, \breve{u})$ must be deleted in line 21 and it holds $\bar{k} \geq k_0$, where k_0 is defined as in Lemma 3.3. Using Lemma 3.4 it follows $x_{\mathrm{act}} \in \varepsilon\text{-}\min(f, X^0) \cap \breve{X}$, $x_{\mathrm{act}} \in A$, $\omega(\breve{X}) \leq \delta$, and hence $\|x_{\mathrm{act}} - \bar{x}\|_2 \leq \delta$ – in contradiction to (13). □

3.2 Notes on the used implementations

For the first numerical tests of the modified αBB method in the following subsection we used Matlab (version R2011b) and four different implementations,

$$\mathbf{mod}\,\boldsymbol{\alpha}^{\mathbf{glob}}\,\mathbf{BB},\ \mathbf{mod}\,\boldsymbol{\alpha}^{\mathbf{loc}}\,\mathbf{BB},\ \mathbf{mod}\,\boldsymbol{\alpha}^{\mathbf{loc}}_{\boldsymbol{i,d=1}}\,\mathbf{BB},\text{ and }\mathbf{mod}\,\boldsymbol{\alpha}^{\mathbf{loc}}_{\boldsymbol{i,d=u-l}}\,\mathbf{BB},$$

which we describe shortly.

To compute a parameter α which satisfies (3) in Lemma 2.4 one possible approach is to apply interval arithmetics and an interval variant of Gerschgorin's theorem (see [10]). Therefor for given $X \in \mathbb{IR}^n$ and $f \in C^2(\mathbb{R}^n, \mathbb{R})$ let $\nabla^2 f(X)_{ij} = [l(\nabla^2 f(X)_{ij}), u(\nabla^2 f(X)_{ij})] \in \mathbb{IR}$ with $\nabla^2 f(X)_{ij} = \nabla^2 f(X)_{ji}$ and $\nabla^2 f(x)_{ij} \in \nabla^2 f(X)_{ij}$ for all $i, j \in \{1, \ldots, n\}$ and all $x \in X$. We define the corresponding (symmetric) interval Hessian matrix $[\nabla^2 f(X)] = (\nabla^2 f(X)_{ij})$ by

$$\left[\nabla^2 f(X)\right] := \left\{H = (h_{ij}) \in \mathbb{R}^{n\times n} \;\middle|\; H = H^\top,\ \forall\, i, j \in \{1, \ldots, n\} :\ h_{ij} \in \nabla^2 f(X)_{ij}\ \right\}$$

and based on this

$$\lambda_{\min}\left(\left[\nabla^2 f(X)\right]\right) := \min\left\{\lambda_{\min}(H) \;\middle|\; H \in \left[\nabla^2 f(X)\right]\right\}.$$

Obviously it holds

$$\min\left\{\lambda_{\min}(\nabla^2 f(x)) \;\middle|\; x \in X\right\} \geq \lambda_{\min}\left(\left[\nabla^2 f(X)\right]\right)$$

and a straightforward extension of Gerschgorin's theorem to interval matrices (see for instance [2, Theorem 3.2]) yields the lower bound

$$\lambda_{\min}\left(\left[\nabla^2 f(X)\right]\right) \geq \min_i \left(l(\nabla^2 f(X)_{ii}) - \sum_{i \neq j} \max\left\{\left|l(\nabla^2 f(X)_{ij})\right|, \left|u(\nabla^2 f(X)_{ij})\right|\right\}\right).$$

Therefore a (simple) possibility to determine a parameter $\alpha = \alpha(X) \in \mathbb{R}_+$ satisfying (3) is given by

$$\alpha(X) := \max\left\{0, -\frac{1}{2}\min_i \left(l(\nabla^2 f(X)_{ii}) - \sum_{i \neq j} \max\left\{\left|l(\nabla^2 f(X)_{ij})\right|, \left|u(\nabla^2 f(X)_{ij})\right|\right\}\right)\right\}. \tag{14}$$

We use in all our implementations INTLAB (version V8, [30]) and automatic differentiation for the computation of the elements $\nabla^2 f(X)_{ij}$ of the interval Hessian matrix $[\nabla^2 f(X)]$. We denote our implementation of the modified αBB method using (14) for the computation of a global parameter $\alpha := \alpha(X^0)$ in line 1 shortly by **mod $\boldsymbol{\alpha}^{\mathbf{glob}}$ BB**.

A local recalculation of the parameter $\alpha = \alpha(\bar{X})$ for every subbox $\bar{X} \in \mathrm{sB}(X^\star, 1)$ in line 6 before the computation of $\bar{x} \in \operatorname{argmin}_{x \in \bar{X}} \Phi_{\alpha,\bar{X}}(x)$ and $\bar{u} = \min_{x \in \bar{X}} \Phi_{\alpha,\bar{X}}(x)$ in line 7 in every iteration $k \geq 1$ (instead of a global calculation of the parameter $\alpha = \alpha(X^0)$) may yield a substantial reduction of the required iteration numbers.

If the parameters $\alpha^\star$ and $\bar{\alpha}$ are chosen such that

$$\begin{aligned} \alpha^\star &\geq \max\left\{0,\; -\tfrac{1}{2}\min\left\{\lambda_{\min}(\nabla^2 f(x)) \;\middle|\; x \in X^\star\right\}\right\}, \\ \bar{\alpha} &\geq \max\left\{0,\; -\tfrac{1}{2}\min\left\{\lambda_{\min}(\nabla^2 f(x)) \;\middle|\; x \in \bar{X}\right\}\right\}, \end{aligned} \tag{15}$$

and $\alpha^\star \geq \bar{\alpha}$, then by [20, Property 5] $\Phi_{\alpha^\star,X^\star}$ is a convex underestimator of f w.r.t. $X^\star$, $\Phi_{\bar{\alpha},\bar{X}}$ is a convex underestimator of f w.r.t. $\bar{X}$, and it holds

$$\min_{x \in X^\star} \Phi_{\alpha^\star,X^\star}(x) \leq \min_{x \in \bar{X}} \Phi_{\bar{\alpha},\bar{X}}(x).$$

In case we determine $\alpha^\star$ and $\bar{\alpha}$ using (14), i.e $\alpha^\star := \alpha(X^\star)$ and $\bar{\alpha} := \alpha(\bar{X})$, then (15) and $\alpha^\star \geq \bar{\alpha}$ are satisfied, if $[\nabla^2 f(\cdot)]$ is inclusion isotonic, i.e. it holds $\nabla^2 f(\bar{X})_{ij} \subset \nabla^2 f(X^\star)_{ij}$ for all $i, j \in \{1, \ldots, n\}$ (see for instance [25]). We denote by **mod $\boldsymbol{\alpha}^{\mathbf{loc}}$ BB** our implementation of the modified αBB method using in every iteration $k \geq 1$ a recalculation of the parameter $\bar{\alpha} := \alpha(\bar{X})$ according to (14) for every subbox $\bar{X} \in \mathrm{sB}(X^\star, 1)$ in line 6.

For the two remaining implementations of the modified αBB method we use instead of the classical approach for a convex underestimator defined by (2) the following slight modification introduced in [5]. Therefor the twice continuously differentiable objective function f will be underestimated on the box $X = [l(X), u(X)] \in \mathbb{R}^n$ by a function $\tilde{\Phi}_{\tilde{\alpha},X} : \mathbb{R}^n \to \mathbb{R}$ defined by

$$\tilde{\Phi}_{\tilde{\alpha},X}(x) := f(x) + \sum_{i=1}^{n} \tilde{\alpha}_i (l(X)_i - x_i)(u(X)_i - x_i)$$

with $\tilde{\alpha}_i \geq 0$ for all $i \in \{1, \ldots, n\}$. Using a so-called Scaled Gerschgorin Theorem (see for instance [2, Theorem 3.13]) it can be shown that the function $\tilde{\Phi}_{\tilde{\alpha},X}$ is a convex underestimator of f w.r.t. X if for a vector $d \in \operatorname{int}(\mathbb{R}^n_+)$ the parameter $\tilde{\alpha} = \tilde{\alpha}(X) \in \mathbb{R}^n_+$ is defined by

$$\tilde{\alpha}_i(X) := \max\left\{0, -\frac{1}{2}\left(l(\nabla^2 f(X)_{ii}) - \sum_{i \neq j} \max\left\{\left|l(\nabla^2 f(X)_{ij})\right|, \left|u(\nabla^2 f(X)_{ij})\right|\right\}\frac{d_j}{d_i}\right)\right\} \tag{16}$$

for all $i \in \{1, \ldots, n\}$.

For both implementations we use a local calculation of the parameter

$$\tilde{\alpha} := \tilde{\alpha}(\bar{X}) = (\tilde{\alpha}_1(\bar{X}), \ldots, \tilde{\alpha}_n(\bar{X}))^\top \in \mathbb{R}^n_+$$

according to (16) for every subbox $\bar{X} \in \mathrm{sB}(X^\star, 1)$ as described above, where we set $d_i := 1$ for all $i \in \{1, \ldots, n\}$ in the case of **mod** $\boldsymbol{\alpha}^{\mathbf{loc}}_{\boldsymbol{i,d=1}}$ **BB** and $d := u(\bar{X}) - l(\bar{X})$ in the case of **mod** $\boldsymbol{\alpha}^{\mathbf{loc}}_{\boldsymbol{i,d=u-l}}$ **BB**, respectively.

Obviously, if we compare the implementations **mod** $\boldsymbol{\alpha}^{\mathbf{loc}}$ **BB** and **mod** $\boldsymbol{\alpha}^{\mathbf{loc}}_{\boldsymbol{i,d=1}}$ **BB** then in every iteration $k \geq 1$ for every subbox $\bar{X} \in \mathrm{sB}(X^\star, 1)$ in line 6 it holds $\bar{\alpha} \geq \tilde{\alpha}_i$ for all $i \in \{1, \ldots, n\}$ and thus $\Phi_{\bar{\alpha},\bar{X}}(x) \leq \tilde{\Phi}_{\tilde{\alpha},\bar{X}}(x)$ for all $x \in \bar{X}$.

For more detailed explanations regarding the used approaches we refer to [2]. We want to note that usually there will be a trade-off between the reduction of the required iteration numbers and the needed CPU-time for the additional local calculations of the parameters. We shortly analyze this effect in the following subsection 3.3.

In all our Matlab-implementations we use for solving the box-constrained local optimizations problems (computation of $\bar{x}$ and $\bar{u}$ in line 7) the SQP algorithm of the solver **fmincon** of the Optimization Toolbox with the default settings. To ensure that the approximations of the solutions are feasible, we demand an exitflag greater than or equal to 1.

Clearly, the magnitude of the parameters α and $\tilde{\alpha}$ have an important influence on the convergence rate of the implementations of the modified αBB method (for instance on the maximal number of iterations in the worst case, see the proof of Theorem 3.6.) Thus the determination of parameters which lead to tight convex underestimators is of vital importance. For further explanations in this direction, regarding to other approaches for convex underestimators, and regarding to other branching strategies we refer for instance to [1, 2, 3, 4, 9, 14, 15, 23, 24, 29, 31, 32, 33].

3.3 First computational results

All experiments of this subsection have been performed with an Intel(R) Core(TM) i3-2105 CPU and with 16 GBytes of RAM (2x DDR3-1333/8 GB), using the operating system Windows 7 Professional. Moreover, let A be the set created by the particular implementation of Algorithm 1. For the presentation of our numerical results in tabular form we use the sets

$$\begin{aligned}
\mathrm{GLOB} &:= \{\bar{x} \in \operatorname{argmin}_{x \in X^0} f(x) \mid \exists\, a \in A : \ a \in (\varepsilon,\delta)\text{-}\min(f, X^0, \bar{x})\}\,, \\
A_{\neg\varepsilon} &:= \{a \in A \;\Big|\; f(a) - \min_{x \in X^0} f(x) > \varepsilon\}, \\
A_{\neg\delta} &:= \{a \in A \mid \forall\, \bar{x} \in \operatorname{argmin}_{x \in X^0} f(x) : \|a - \bar{x}\|_2 > \delta\},
\end{aligned}$$

and the following notations :

abbreviation	denotation
iter	number of required iterations
CPU	required CPU time in seconds
$\lvert A\rvert$	cardinality of the set A (created by the algorithm)
$\lvert\,\mathrm{GLOB}\,\rvert$	cardinality of the set GLOB
$\lvert A_{\neg\varepsilon}\rvert$	cardinality of the set $A_{\neg\varepsilon}$
$\lvert A_{\neg\delta}\rvert$	cardinality of the set $A_{\neg\delta}$
$-$	The algorithm does not terminate after 100 000 iterations or 10 hours of CPU time.

Obviously in the case $|\operatorname{argmin}_{x \in X^0} f(x)| < \infty$ the set A created by our implementations of the modified αBB method is an (ε, δ)-minimal set of f w.r.t. X^0 if and only if

$$|\,\mathrm{GLOB}\,| = |\operatorname{argmin}_{x \in X^0} f(x)| \quad \text{and} \quad |A_{\neg\varepsilon}| = 0.$$

At first we consider the box-constrained optimization problem (1) for the following four classical test problems

	$f : \mathbb{R}^2 \to \mathbb{R}$ with
Rastrigin [27]	$f(x) := 20 + x_1^2 + x_2^2 - 10\,(\cos(2\pi x_1) + \cos(2\pi x_2))$
Easom [8]	$f(x) := -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2 - (x_2-\pi)^2\right)$
Branin [7]	$f(x) := \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$
Levy No.3 [16]	$f(x) := \sum\limits_{i=1}^{5}\left[i\cos\left((i+1)x_1 + i\right)\right]\sum\limits_{j=1}^{5}\left[j\cos\left((j+1)x_2 + j\right)\right]$

with:

	X^0	$\operatorname{argmin}_{x \in X^0} f(x)$	$\min_{x \in X^0} f(x)$
Rastrigin	$\left[\begin{pmatrix}-5.12\\-5.12\end{pmatrix}, \begin{pmatrix}5.12\\5.12\end{pmatrix}\right]$	$\left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}$	0
Easom	$\left[\begin{pmatrix}-100\\-100\end{pmatrix}, \begin{pmatrix}100\\100\end{pmatrix}\right]$	$\left\{\begin{pmatrix}\pi\\\pi\end{pmatrix}\right\}$	-1
Branin	$\left[\begin{pmatrix}-5\\0\end{pmatrix}, \begin{pmatrix}10\\15\end{pmatrix}\right]$	$\left\{\begin{pmatrix}-\pi\\\frac{491}{40}\end{pmatrix}, \begin{pmatrix}\pi\\\frac{91}{40}\end{pmatrix}, \begin{pmatrix}3\pi\\\frac{99}{40}\end{pmatrix}\right\}$	$\frac{5}{4\pi} \approx 0.397887$
Levy No.3	$\left[\begin{pmatrix}-10\\-10\end{pmatrix}, \begin{pmatrix}10\\10\end{pmatrix}\right]$	18 global minima	≈ -186.730909

Clearly, the numerical solutions of all box-constrained convex optimization problems in line 7 are in general not exact. For this reason we replace in all implementations the condition $\bar{u} \leq v_{\text{glob}}$ in line 8 by

$$\bar{u} \leq v_{\text{glob}} + \varepsilon_{\text{save}}$$

for a fixed $\varepsilon_{\text{save}} \geq 0$. Table 1 presents our numerical results for the above defined four test problems by using the parameters $\varepsilon := 1\text{E}-3$, $\delta := 1\text{E}-1$, and $\varepsilon_{\text{save}} := 1\text{E}-6$.

Table 1: Numerical results for $\varepsilon := 1\text{E}-3$, $\delta := 1\text{E}-1$, and $\varepsilon_{\text{save}} := 1\text{E}-6$.

	mod $\boldsymbol{\alpha}^{\textbf{glob}}$ BB					
	iter	CPU	$\lvert A\rvert$	$\lvert\,\text{GLOB}\,\rvert$	$\lvert A_{\neg\varepsilon}\rvert$	$\lvert A_{\neg\delta}\rvert$
Rastrigin	766	20	4	1	0	0
Easom	-	-	-	-	-	-
Branin	821	20	47	3	0	0
Levy No.3	10458	324	132	18	0	0
	mod $\boldsymbol{\alpha}^{\textbf{loc}}$ BB					
	iter	CPU	$\lvert A\rvert$	$\lvert\,\text{GLOB}\,\rvert$	$\lvert A_{\neg\varepsilon}\rvert$	$\lvert A_{\neg\delta}\rvert$
Rastrigin	641	34	4	1	0	0
Easom	86	4	1	1	0	0
Branin	112	5	6	3	0	0
Levy No.3	4305	523	18	18	0	0
	mod $\boldsymbol{\alpha}^{\textbf{loc}}_{\boldsymbol{i,d=1}}$ BB					
	iter	CPU	$\lvert A\rvert$	$\lvert\,\text{GLOB}\,\rvert$	$\lvert A_{\neg\varepsilon}\rvert$	$\lvert A_{\neg\delta}\rvert$
Rastrigin	580	31	4	1	0	0
Easom	80	4	1	1	0	0
Branin	91	4	4	3	0	0
Levy No.3	4289	514	18	18	0	0
	mod $\boldsymbol{\alpha}^{\textbf{loc}}_{\boldsymbol{i,d=u-l}}$ BB					
	iter	CPU	$\lvert A\rvert$	$\lvert\,\text{GLOB}\,\rvert$	$\lvert A_{\neg\varepsilon}\rvert$	$\lvert A_{\neg\delta}\rvert$
Rastrigin	580	31	4	1	0	0
Easom	80	4	1	1	0	0
Branin	77	3	3	3	0	0
Levy No.3	4277	513	18	18	0	0

Only **mod $\boldsymbol{\alpha}^{\textbf{glob}}$ BB** for the test problem **Easom** does not terminate after 100 000 iterations. No created subbox of X^0 was deleted from the list $\mathcal{L}$ in the algorithm based on our bounding criteria. The reason for that may be the very large value of $\alpha(X^0)$ (defined according to (14)) and the related strong underestimation by the convex underestimators for all subproblems:

	Rastrigin	**Easom**	**Branin**	**Levy No.3**
$\alpha(X^0)$	$1.963921\text{E}+002$	$4.296532\text{E}+004$	$1.698258\text{E}+001$	$5.075000\text{E}+003$

All algorithms determine for all considered test problems an (ε, δ)-minimal set A of f w.r.t. X^0. Moreover, for all considered problems it holds $|A_{\neg\delta}| = 0$ and for all problems except **Rastrigin** the cardinalities of the sets A decrease if a local recalculation of the parameters $\alpha = \alpha(\bar{X})$ or $\tilde{\alpha} = \tilde{\alpha}(\bar{X})$ is used. In these cases (local recalculation) it holds $|A| - |\operatorname{argmin}_{x \in X^0} f(x)| \leq 3$ and for $\textbf{mod}\, \boldsymbol{\alpha}^{\textbf{loc}}_{\boldsymbol{i,d=u-l}}\, \textbf{BB}$ except **Rastrigin** even $|A| = |\operatorname{argmin}_{x \in X^0} f(x)|$.

The mentioned trade-off between the reduction of the required iteration number and the needed CPU-time for the local recalculations can be stated only for the test problems **Rastrigin** and **Levy No.3**. For the test problems **Easom** and **Branin** a significant decrease of the required iteration number as well as of the needed CPU-time turn out. We want to note that for $\textbf{mod}\, \boldsymbol{\alpha}^{\textbf{glob}}\, \textbf{BB}$ and the test problem **Rastrigin** the four elements of A are all equal to the unique minimal point. This is due to the fact that in this case the globally minimal point is at the boundary of four created subboxes and also the solution of all the corresponding box-constrained convex optimization problems.

Tables 2 and 3 illustrate the importance of the choice of $\varepsilon_{\text{save}}$. For the algorithm $\textbf{mod}\, \boldsymbol{\alpha}^{\textbf{glob}}\, \textbf{BB}$ and the setting $\varepsilon_{\text{save}} := 0$ instead of $\varepsilon_{\text{save}} := 1\text{E} - 6$ no influences can be observed. However, all algorithms with a local recalculation and $\varepsilon_{\text{save}} = 0$ determine only for the test problem **Rastrigin** an (ε, δ)-minimal set A of f w.r.t. X^0. Table 2 shows this fact for $\textbf{mod}\, \boldsymbol{\alpha}^{\textbf{loc}}\, \textbf{BB}$. Note that for the test problems **Easom** and **Levy No.3** it holds $|A| = 0$. Moreover, an increase of the accuracy of the SQP algorithm used in **fmincon** by choosing the parameters TolX := eps, TolFun := eps, and TolCon := eps does not change this issue.

Table 2: Numerical results for $\varepsilon := 1\text{E} - 3$, $\delta := 1\text{E} - 1$, and $\varepsilon_{\text{save}} := 0$.

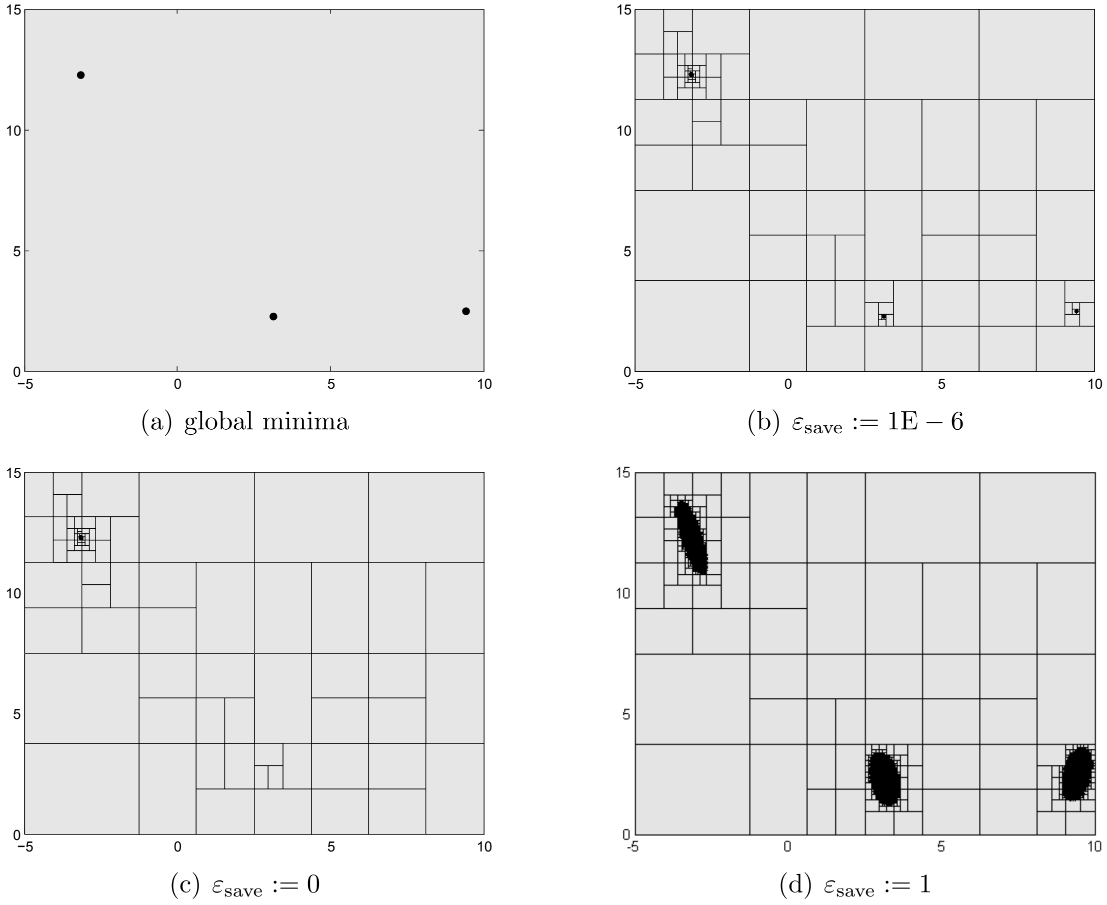
	$\textbf{mod}\, \boldsymbol{\alpha}^{\textbf{loc}}\, \textbf{BB}$					
	iter	CPU	$\|A\|$	$\|$ GLOB $\|$	$\|A_{\neg\varepsilon}\|$	$\|A_{\neg\delta}\|$
Rastrigin	641	34	4	1	0	0
Easom	81	4	0	0	0	0
Branin	94	4	4	1	0	0
Levy No.3	4251	511	0	0	0	0

Clearly, a choice of a too large value of $\varepsilon_{\text{save}}$ for the implementations with a local recalculation may yield a substantial increase of the required iteration number and/or the needed CPU-time. Table 3 illustrates this fact for $\textbf{mod}\, \boldsymbol{\alpha}^{\textbf{loc}}\, \textbf{BB}$ and $\varepsilon_{\text{save}} := 1$ (see the test problem **Branin**). For the test problem **Easom** the algorithm does not terminate within 100 000 iterations and for all remaining test problems no (ε, δ)-minimal set A of f w.r.t. X^0 is determined ($|A_{\neg\varepsilon}| \neq 0$).

Table 3: Numerical results for $\varepsilon := 1\mathrm{E}-3$, $\delta := 1\mathrm{E}-1$, and $\varepsilon_{\text{save}} := 1$.

	mod $\boldsymbol{\alpha}^{\text{loc}}$ BB					
	iter	CPU	$\|A\|$	$\|\text{GLOB}\|$	$\|A_{\neg\varepsilon}\|$	$\|A_{\neg\delta}\|$
Rastrigin	699	37	16	1	12	10
Easom	-	-	-	-	-	-
Branin	1840	163	1548	3	1532	1484
Levy No.3	4385	532	58	18	40	0

Figure 1 illustrates the previous results for the implementation **mod $\boldsymbol{\alpha}^{\text{loc}}$ BB**, the test problem **Branin**, and the considered values of $\varepsilon_{\text{save}}$:



(a) global minima

(b) $\varepsilon_{\text{save}} := 1\mathrm{E}-6$

(c) $\varepsilon_{\text{save}} := 0$

(d) $\varepsilon_{\text{save}} := 1$

Figure 1: Subboxes and sets A for the implementation **mod $\boldsymbol{\alpha}^{\text{loc}}$ BB** using the test problem **Branin**, $\varepsilon := 1\mathrm{E}-3$, and $\delta := 1\mathrm{E}-1$.

We do not require for our modified αBB method that all global minima of the box-constrained optimization problem are isolated or in the interior of the box X^0. For numerical tests in the case of an infinite number of global minima and/or global minima at the boundary of X^0 we use the following three test problems

	$f : \mathbb{R}^2 \to \mathbb{R}$ with
Test01	$f(x) := \left(\frac{x_1^2}{4^2} + \frac{x_2^2}{2^2} - 1\right)^2$
Test02	$f(x) := \frac{1}{10}\left(x_1(1-x_2) + x_2(1-x_1)\right)^2$
Test03	$f(x) := \sin^2\left(\frac{5}{4}x_1 + x_2 - 3\right)$
Test04	$f(x) := (x_1 + \sin^2(x_1))\cos^2(x_2)$

with

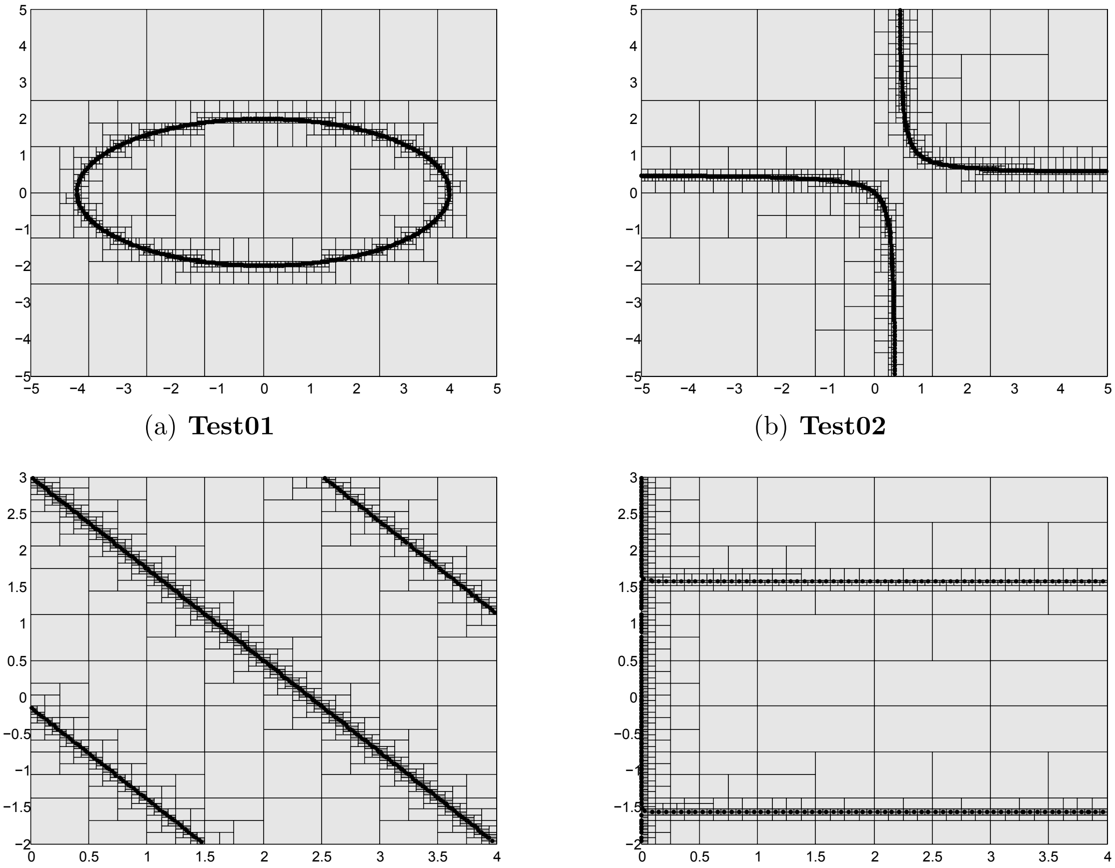
	X^0	$\operatorname{argmin}_{x \in X^0} f(x)$
Test01	$\left[\begin{pmatrix}-5\\-5\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}\right]$	$\left\{\begin{pmatrix}x_1\\x_2\end{pmatrix} \,\middle\vert\, \frac{x_1^2}{4^2} + \frac{x_2^2}{2^2} = 1\right\}$
Test02	$\left[\begin{pmatrix}-5\\-5\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}\right]$	$\left\{\begin{pmatrix}x_1\\x_2\end{pmatrix} \,\middle\vert\, \begin{array}{l} x_1 \in [-5,5] \setminus \left\{\frac{1}{2}\right\}, \\ x_2 = -\frac{x_1}{1-2x_1} \end{array}\right\}$
Test03	$\left[\begin{pmatrix}0\\-2\end{pmatrix}, \begin{pmatrix}4\\3\end{pmatrix}\right]$	$\left\{\begin{pmatrix}x_1\\x_2\end{pmatrix} \,\middle\vert\, \begin{array}{l} \frac{5}{4}x^1 + x_2 = 3 + a, \\ a \in \{-\pi, 0, \pi\} \end{array}\right\} \cap X^0$
Test04	$\left[\begin{pmatrix}0\\-2\end{pmatrix}, \begin{pmatrix}4\\3\end{pmatrix}\right]$	$\left\{\begin{pmatrix}0\\x_2\end{pmatrix} \,\middle\vert\, x_2 \in [-2,3]\right\} \cup \left\{\begin{pmatrix}x_1\\x_2\end{pmatrix} \,\middle\vert\, \begin{array}{l} x_1 \in [0,4], \\ x_2 \in \left\{-\frac{\pi}{2}, \frac{\pi}{2}\right\} \end{array}\right\}$

and the global minimal value $\min_{x \in X^0} f(x) = 0$. Table 4 shows the results of all implementations for the above defined test problems. One can see that the implementations with a local recalculation of the parameters α or $\tilde{\alpha}$ are far better for these test problems. There are no significant differences between these three implementations.

Table 4: Numerical results for the test problems **Test01**, **Test02**, **Test03**, and **Test04** using $\varepsilon := 1\text{E} - 3$, $\delta := 1\text{E} - 1$, and $\varepsilon_{\text{save}} := 1\text{E} - 6$.

	mod $\boldsymbol{\alpha}^{\mathbf{glob}}$ BB			**mod $\boldsymbol{\alpha}^{\mathbf{loc}}$ BB**		
	iter	CPU	$\lvert A\rvert$	iter	CPU	$\lvert A\rvert$
Test01	27783	22878	10875	1267	103	554
Test02	-	-	-	1130	68	437
Test03	5353	1185	2483	969	57	395
Test04	13118	7465	6468	676	40	315
	mod $\boldsymbol{\alpha}^{\mathbf{loc}}_{\boldsymbol{i,d=1}}$ BB			**mod $\boldsymbol{\alpha}^{\mathbf{loc}}_{\boldsymbol{i,d=u-l}}$ BB**		
	iter	CPU	$\lvert A\rvert$	iter	CPU	$\lvert A\rvert$
Test01	1263	98	554	1271	101	562
Test02	1100	71	437	1093	69	433
Test03	963	58	332	963	54	332
Test04	672	40	315	671	39	315

Figure 2 illustrates the previous results for the implementation **mod $\boldsymbol{\alpha}^{\mathbf{loc}}$ BB** and the test problems **Test01**, **Test02**, **Test03**, and **Test04**:



(a) **Test01**

(b) **Test02**

(c) **Test03**

(d) **Test04**

Figure 2: Subboxes and sets A for the implementation $\mathbf{mod}\,\boldsymbol{\alpha}^{\mathbf{loc}}\,\mathbf{BB}$ using $\varepsilon := 1\mathrm{E}-3$, $\delta := 1\mathrm{E}-1$, and $\varepsilon_{\text{save}} := 1\mathrm{E}-6$.

4 An application to robot design

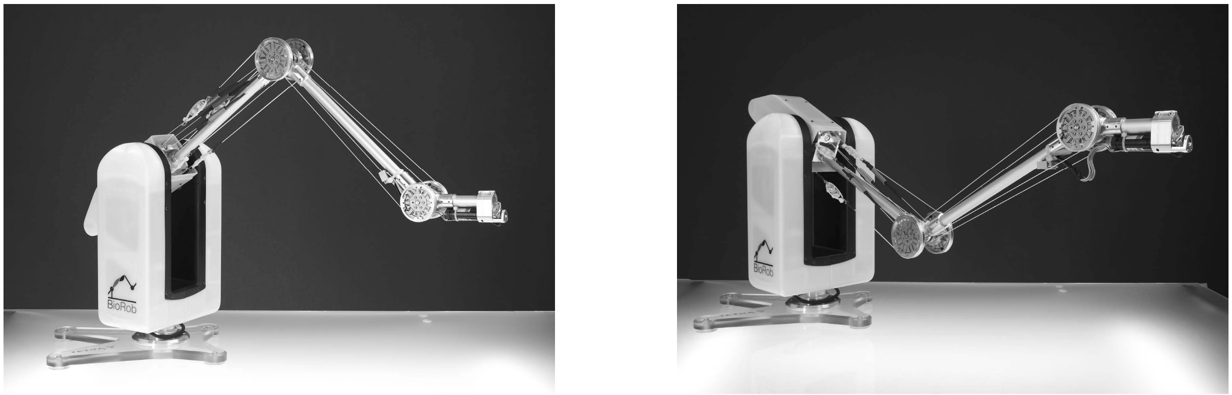
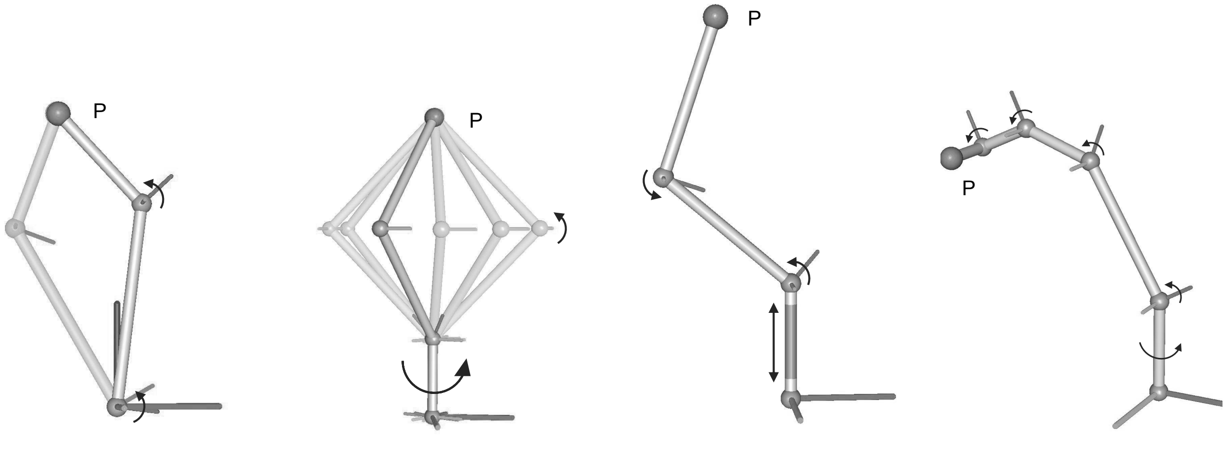


Figure 3: Developmental sample of a robotic arm, with the kind permission of TETRA GmbH Ilmenau, Dr. Andreas Karguth.

In this section, we apply the modified αBB method to a design problem in the field of robotics. Thereby the task is to evaluate which positions $P \in \mathbb{R}^3$ are reachable by the tip of a robotic arm (also called the Tool Center Point (TCP) of the robot) – and which settings of the robotic arm allow to reach this position. The robotic arm is assumed to form a kinematic chain which consists of k arm sections (thrust axes), where l sections have adjustable length and $k-l$ sections have a fixed length. The sections are connected by k joints, where at m joints a specific rotation angle can be chosen and $k-m$ joints have a fixed adjustment. We denote a specific choice of the l adjustable lengths of the arm sections and of the m adjustable angles of the joints as a setting $x \in \mathbb{R}^n$ with $n := l + m$. For an illustration see Figure 4 and for more detailed information we refer to [26].



(a) $n = 2$ (b) $n = 2$ (c) $n = 3$ (d) $n = 5$

Figure 4: Settings of a robotic arm to reach the point P marked with a ball for a robotic arm with (a) two arm sections with fixed length ($l = 0$) and two variable joints ($m = 2$), (b) three arm sections with fixed length ($l = 0$) and two variable joints ($m = 2$), (c) three arm sections, where one is adjustable in the length ($l = 1$), and two variable joints ($m = 2$), (d) five arm sections with fixed length ($l = 0$) and five variable joints ($m = 5$).

For a setting $x \in \mathbb{R}^n$ the position of the TCP can be calculated by evaluating the function $g : \mathbb{R}^n \to \mathbb{R}^3$ with

$$g(x) := \sum_{j=1}^{k} D_j(x) \cdot \begin{pmatrix} 0 \\ l_j(x) \\ 0 \end{pmatrix} \tag{17}$$

(direct kinematics), where $D_j(x)$ is the product of 3×3 rotation matrices and $l_j(x)$ denotes the length of the arm section j for the current setting x.

We are interested in the problem of inverse kinematics which means to find to a given point $P \in \mathbb{R}^3$ a setting $x \in \mathbb{R}^n$ such that $\|g(x) - P\|_2^2 = 0$. It is important to know all possible settings. The reason is that in general the robotic arm has to perform several movements in a row. So it is advantageous to choose for the current position a setting which gives a good start for the next setting or which is in some sense stable (not too small or too large angles).

Moreover, from a practical perspective there are bounds on the possible settings x_i, $i = 1, \ldots, n$, which results in a box $X^0 \in \mathbb{R}^n$ and the constraint $x \in X^0$. Therefore, we search for all globally optimal solutions of the nonlinear non-convex optimization problem

$$\min_{x \in X^0} f(x) \quad \text{with} \quad f(x) := \|g(x) - P\|_2^2. \tag{18}$$

In case the globally optimal value is zero, then all globally optimal solutions define a possible setting for the robotic arm to reach the position P. This might be no, one, a finite number, or an infinite number of settings. The objective function of the optimization problem (18) is twice continuously differentiable and we are able to apply the modified αBB method as discussed in section 3.

All numerical calculations of this section have been performed with an Intel(R) Core(TM) i5-4200 CPU and with 8 GBytes of RAM (2x DDR3-800/4 GB), using the operating system WINDOWS 8.1. Moreover, we use a C++ implementation of **mod $\boldsymbol{\alpha}^{\mathbf{loc}}$ BB** where we choose for the parameters of the algorithm $\varepsilon := 1\text{E} - 5$, $\delta := 1.0$, and $\varepsilon_{\text{save}} := 1\text{E} - 6$. This is, on the one hand, based on heuristic examinations. On the other hand, from the practical point of view, it is clear that small differences in the settings cannot be realized. After applying the modified αBB method which results in an (ε, δ)-minimal set A, we reduce the large number of points in A by clustering those which differ in no angle by more than 1° and in no thrust axis by more than 0.02. We give only one combined solution for each of these clusters. In the following example we give some results for the robotic arms of Figure 4.

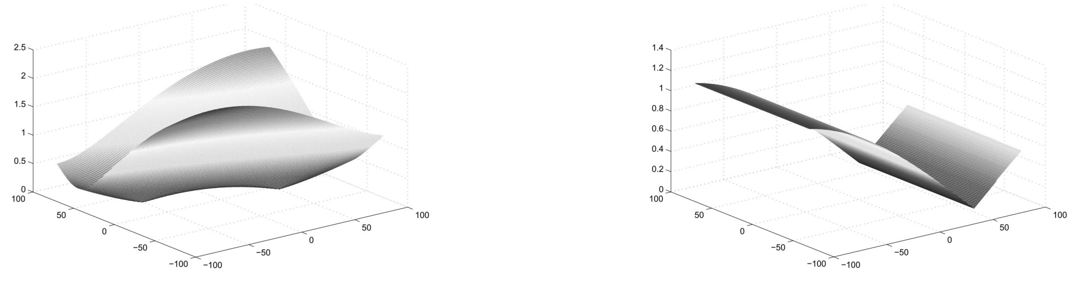
Example 4.1.

(a) *For the robotic arm of Figure 4(a) with $n = 2$ we have $X^0 := \left[\begin{pmatrix}-100\\-100\end{pmatrix}, \begin{pmatrix}100\\100\end{pmatrix}\right]$ as the angles are assumed to be bounded by ±100°. The root of the objective function f of the optimization problem (18) is shown in Figure 5(a). In this case two settings allow to reach the point P with the TCP. These two globally optimal solutions of the optimization problem (18) were approximated by two ε-minimal points after the clustering in 0.13 seconds. An earlier* MATLAB *implementation required instead 34.2 seconds.*

(b) *For the robotic arm of Figure 4(b) with $n = 2$ we have $X^0 := \left[\begin{pmatrix}0\\-100\end{pmatrix}, \begin{pmatrix}50\\0\end{pmatrix}\right]$. Again, the root of the objective function f is shown in Figure 5(b). Due to the specific position of P any choice of the first angle x_1 can be compensated by x_2 such that the TCP still reaches the point P. Therefore we have an infinite number of globally optimal solutions of the optimization problem (18). The modified αBB method as described above delivers after the clustering 31 ε-minimal points in 0.32 seconds.*

(c) *For the robotic arm of Figure 4(c) with $n = 3$ we have $X^0 := \left[\begin{pmatrix}0.2\\20\\-70\end{pmatrix}, \begin{pmatrix}1.5\\60\\-30\end{pmatrix}\right]$, where the length of the first arm section is adjustable between* 0.2 *and* 1.5*. In this case there is an infinite number of globally optimal solutions of (18). After clustering the method delivers 30 ε-minimal points in 2.43 seconds.*

(d) *We consider the robotic arm of Figure 4(d) with $n = 5$. The box X^0 is determined by lower and upper bounds for the angles which allow the angles to vary in (five different) intervals of length* 10*. There is again an infinite number of globally optimal solutions of (18). The modified αBB method delivers after the clustering 76 ε-minimal points in around 2.5 hours. We recalculated this example again with a larger feasible set X^0 (all intervals for the angles having length* 20*). In this case the algorithm did not stop within three days.*



(a)

(b)

Figure 5: The roots of the objective functions of the robotic arm of Example 4.1(a) and (b)

The above examples also illustrate an effect which is well-known from the classical αBB method: the required calculation time depends on the dimension of the search space (i.e. on n). Moreover, the modified αBB method is rather sensitive with respect to the choice of the parameters ε and δ in relation to the length of the box X^0.

5 Outlook

Our modified αBB-method for box constrained optimization problems can be extended to general constrained problems where the feasible set is within a box X^0 using the same techniques as known for the classical αBB method. For convex inequality constraints the techniques are quite straightforward, as more or less just the boxes X of the subproblems have to be replaced by the intersection of X with the convex feasible sets. Subboxes X which contain no feasible point are discarded from further examinations. For non-convex inequality constraints, convex underestimators have to be calculated. In this case, additional difficulties arise as considered points x which are feasible for the relaxed problems may not be feasible for the original problem. For more details we refer for instance to [9, Chapter 12]. Moreover, the performance of the algorithm could also be improved by incorporating additional local information (monotonicity test, concavity test, and interval Newton Gauss-Seidel step) as proposed for instance in [11, 13].

References

[1] Adjiman, C.S., Androulakis, I.P., and Floudas, C.A.: A global optimization method, αBB, for general twice-differeentiable constrained NLPs — II. Implementation and computational results, Computers Che. Engng., 22(9), 1159-1179 (1998)

[2] Adjiman, C.S., Dallwig S., Floudas, C.A., and Neumaier, A.: A global optimization method, αBB, for general twice-differeentiable constrained NLPs — I. Theoretical Advances, Computers Che. Engng., 22(9), 1137-1158 (1998)

[3] Androulakis, I.P. and Floudas, C.A.: Computational Experience with a New Class of Convex Underestimators: Box-constrained NLP problems, J. Global Optim., 29, 249-264 (2004)

[4] Androulakis, I.P. and Floudas, C.A.: A New Class of Improved Convex Underestimators for Twice Continuously Differentiable Constrained NLPs, J. Global Optim., 30, 367-390 (2004)

[5] Androulakis, I.P., Maranas, C.D., and Floudas, C.A.: αBB: A Global Optimization Method for General Constrained Nonconvex Problems, J. Global Optim., 7, 337-363 (1995)

[6] Al-Khayyal, F.A. and Falk, J.E.: Jointly constrained biconvex programming, Maths Oper. Res., 8, 273-286 (1983)

[7] Branin, F.H. and Hoo, S.K.: A method for finding multiple extrema of a function of n variables, in: Lootsma, F. A. (ed.): Numerical Methods of Nonlinear Optimization, Academic Press, London, 231-237 (1972)

[8] Easom, E.E.: A survey of global optimization techniques. M. Eng. thesis, Univ. Louisville, Louisville, KY (1990)

[9] Floudas, C.A.: Deterministic Global Optimization, Kluwer, Dordrecht (2000)

[10] Gerschgorin, S.: Über die Abgrenzung der Eigenwerte einer Matrix, Izv. Akad. Nauk SSSR, Ser, fiz. mat., 6, 749-754 (1931)

[11] Hammer, R., Hocks, M., Kulisch, U., and Ratz, D.: C++ Toolbox for Verified Computing I, Springer, Berlin (1995)

[12] Hansen, E.: Global Optimization Using Interval Analysis-The Multi-Dimensional Case, Numer. Math., 34, 247-270 (1980)

[13] Hansen, E. and Walster, G.W.: Global optimization using interval analysis, 2nd ed., revised and expanded, Marcel Dekker, New York (2004)

[14] Hertz, D.: The extreme eigenvalues and stability of real symmetric interval matrices, IEEE Trans. Autom. Cont., 37, 532-535 (1992)

[15] Hladík, M., Daney, D., and Tsigaridas, E.: Bounds on real eigenvalues and singular values of interval matrices, SIAM J. Matrix Anal. Appl., 31(4), 2116-2129 (2010)

[16] Levy, A., Montalvo A., Gomez S., and Galderon A.: Topics in Global Optimization, Springer, New York (1981)

[17] Liu, W.B. and Floudas, C.A.: A remark on the GOP algorithm for global optimization, J. Global Optim., 3, 519-521 (1993)

[18] Maranas, C.D. and Floudas, C.A.: A global optimization approach for Lennard-Jones microclusters, J. Chem. Phys., 97(10), 7667-7678 (1992)

[19] Maranas, C.D. and Floudas, C.A.: A deterministic global optimization approach for molecular structure determination, J. Chem. Phys., 100(2), 1247-1261 (1994)

[20] Maranas, C.D. and Floudas, C.A.: Global Minimum Potential Energy Conformations of Small Molecules, J. Global Optim., 4, 135-170 (1994)

[21] Maranas, C.D. and Floudas, C.A.: Finding all solutions of nonlinearly constrained systems of equitations, J. Global Optim., 7, 143-183 (1995)

[22] McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems, Math. Programming, 10, 147-175 (1976)

[23] Mönningmann, M.: Efficient calculations of bounds on spectra of Hessian matrices, SIAM J. Sci. Comput., 30, 2340-2357 (2008)

[24] Mönningmann, M.: Fast calculations of spectral bounds for Hessian matrices on hyperrectangles, SIAM J. Matrix Anal. Appl., 32(4), 1351-1366 (2011)

[25] Moore, R.E., Kearfott, R.B., and Cloud, M.J.: Introduction to Interval Analysis, SIAM, Philadelphia, PA (2009)

[26] Paul, R. P.: Robot Manipulators: Mathematics, Programming, and Control , MIT Press, Cambridge, MA (1982)

[27] Rastrigin, L. A.: Systems of extremal control, Nauka, Moscow (1974)

[28] Ratz, D.: Automatische Ergebnisverifikation bei globalen Optimierungsproblemen, Diss., Univ. Karlsruhe (1992)

[29] Rohn, J.: Positive definiteness and stability of interval matrices, SIAM J. Matrix Anal. Appl., 15(1), 175-184 (1994)

[30] Rump, S.M.: INTLAB - INTerval LABoratory, in: Csendes, T. (ed.): Developments in Reliable Computing, Kluwer Academic Publishers, Dordrecht, 77-104 (1999)

[31] Schulze Darup, M., Kastsian, M., Mross, S., and Mönningmann, M.: Efficient computation of spectral bounds for Hessian matrices on hyperrectangles for global optimization, J. Global Optim., 58, 631-652 (2014)

[32] Skjäl, A., Westerlund, T., Misener R., and Floudas, C.A.: A generalization of the classical αBB convex underestimation via diagonal and non-diagonal quadratic terms, J. Optim. Theory Appl., 154(2), 462-490 (2012)

[33] Skjäl, A. and Westerlund, T.: New methods for calculating αBB-type underestimators, J. Global Optim., 58, 411-427 (2014)